PERENCANAAN DAN PEMBUATAN INTERFACE PENGONTROL PINTU OTOMATIS UNTUK KARYAWAN DENGAN TEKNIK FREQUENCY SHIFT KEYING

Disusun oleh :

NURLAMBANG ATMOJO

NRP : 2912201782

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

1995

PERENCANAAN DAN PEMBUATAN INTERFACE PENGONTROL PINTU OTOMATIS UNTUK KARYAWAN DENGAN TEKNIK FREQUENCY SHIFT KEYING

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik Elektro
Pada
Bidang Studi Teknik Sistem Komputer
Jurusan Teknik Elektro
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya

Mengetahui / Menyetujui

Dosen Pembimbing I	Dosen Pembimbing II
(Ir. YOYON K. SUPRAPTO, M.Sc.)	(Ir. HANNY B. NUGROHO)
NIP. 130 687 439	NIP. 131 651 433

SURABAYA
SEPTEMBER, 1995

ABSTRAK

Dalam industri, selain **manufacturing** ada dua hal lain yang penting, yaitu **controlling** dan **planning**. Planning akan menjalankan controlling, controlling akan mengatur pelaksanaan manufacturing, sedang hasil akhir manufacturing akan kembali ke planning. Kebanyakan dalam hal controlling yang diatur adalah sumber daya manusia.

Tugas akhir ini sebenarnya suatu **upaya untuk menguasai teknologi**. Dalam hal ini teknologi untuk pengontrollan pintu di suatu pabrik yang besar dengan menggunakan perantaraan teknologi Frekwensi Shift Keying.

Beberapa keuntungan dapat dicapai, seperti mengurangi beban tugas pengawas / SATPAM, pusat pengontrollan tidak perlu harus didekat pintu, tercatatnya kapan waktu seseorang mengakses pintu, dan database yang berbasis DBF. Ini memudahkan pengembangan selanjutnya, misalnya kearah absensi, gaji, dan seterusnya.

Tidak mudah memang menghadapi, bahkan menguasai suatu teknologi. Walaupun demikian, dengan segala keterbatasan yang ada, tekad menguasai teknologi, yang seakan terus menantang, tidak akan surut.

KATA PENGANTAR

Syukur Alhamdullilah kepada Tuhan Yang Maha Esa yang telah memberikan berkahNya sehingga Tugas Akhir ini dapat diselesaikan.

Tugas Akhir ini yang berjudul :

"PERANCANGAN DAN PEMBUATAN INTERFACE PENGONTROL PINTU OTOMATIS UNTUK KARYAWAN DENGAN TEKNIK FSK"

dilaksanakan guna memenuhi satuan kredit semester yang merupakan persyaratan dalam menyelesaikan pendidikan Strata 1 (Satu) di Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember, Surabaya.

Semoga Tugas Akhir ini bermanfaat dan dapat dijadikan dasar untuk pengembangan selanjutnya, terutama antar disiplin ilmu dan teknologi, yang semakin lama semakin terasa berintegrasi. Dengan harapan pula semoga akan meningkatkan penguasaan teknologi di masyarakat, terutama dalam lingkungan Almamater tercinta.

Surabaya, Agustus 1995

Penyusun

UCAPAN TERIMAKASIH

Banyak pihak telah memberikan bantuan kepada penyusun dalam menyelesaikan Tugas Akhir ini. Karena itu pada kesempatan ini penyusun mengucapkan terima kasih kepada :

- Ayah, ibu dan saudara-saudara penulis, yang dengan semangat dan dukungannya telah sangat membantu penulis untuk menyelesaikan Tugas Akhir ini.
- Bapak Ir. Sallehudin, selaku Ketua Jurusan Teknik Elektro FTI ITS.
- Bapak Ir. Yoyon K. Suprapto M.Sc., selaku Koordinator Bidang Studi Komputer Jurusan Teknik Elektro FTI ITS, sekaligus sebagai dosen pembimbing I.
- Bapak Ir. Zainal Alim, selaku dosen wali penulis.
- Bapak Ir. Hanny B. Nugroho, selaku dosen pembimbing II.
- Terimakasih pula kepada Dosen Bidang Studi Komputer lain , Ir. Hardiristanto M.Sc., Bp. Ir Eko, atas atensinya, Bp. Mardi, terimakasih atas saran-sarannya, dan Pak Ketut, untuk dukungannya juga, Ca-Dos Pak Hari, Eko, Mbak Susi, dan Pak Sigit, semoga sukses.

- Seluruh rekan di Lab. B-201, Hari(Endhut), Akok, Gan(lt), Phink, Tiwul, Mboss, Ghoank, Rini, K-pet, Borrox, Ithenk, Ollie, Yuda, Bull, Odhont, We&u, Chris, Haho, Sidik, Able, Tokechang, Tompel, Taegh, dan lainnya.
- Rekan-rekan dari Lab. lain, Lexi, Wasiq, Agung, Firda, Imam, dll.
- Seluruh Dosen dan staf beserta karyawan Jurusan Teknik Elektro, Paimin, Pak Mo, dll.
- Dan juga seluruh rekan-rekan mahasiswa Jurusan Teknik Elektro, FTI-ITS.

Semoga Tugas Akhir ini dapat memberikan semangat dalam usaha penguasaan ilmu pengetahuan dan teknologi, selain itu juga dapat memberikan sumbangan pada iptek itu sendiri.

Surabaya, Agustus 1995

Penyusun

DAFTAR ISI

JUDUL i
PENGESAHAN ii
ABSTRAK iii
KATA PENGANTAR iv
DAFTAR ISI vii
DAFTAR GAMBAR xi
DAFTAR TABEL xiii

BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Permasalahan 2
1.3 Tujuan 3
1.4 Pembatasan Masalah 4
1.5 Metodologi 4
1.6 Sistematika Penulisan 5
1.7 Relevansi 6

BAB II TEORI PENUNJANG 8
2.1 PENGONTROL PINTU UNTUK KARYAWAN 8
2.2 STRUKTUR DATA 10
2.3 TIPE DATA 10
2.3.1 Tipe Data Dasar 11
2.3.2 Array 12
2.3.3 Structure 13

2.3.4 Linked List 13
2.3.5 File 14
2.4 BASIS DATA 14
2.4.1 Field Dan Record 16
2.4.2 Tag, Index Dan Filter 17
2.5 KOMUNIKASI DATA 18
2.5.1 Komunikasi Data Paralel 18
2.5.2 Komunikasi Data Serial 19
2.6 MODEM 21
2.6.1 Dasar Modem 21
2.6.2 Baud Rate 24
2.7 MIKROKONTROLER 8031 27
2.7.1 Arsitektur Dan Organisasi 8031 28
2.7.2 Fungsi Pin-Pin 8031 30
2.7.3 Perangkat Keras CPU 33
2.7.4 Unit ALU 38
2.7.5 Rangkaian Osilator 39
2.7.6 Pewaktu CPU 40
2.7.7 Memori 41
2.7.7.1 Organisasi Memori 41
2.7.7.2 Mode-Mode Pengalamatan 46
2.7.8 Operasi Input/Output 46
2.7.9 Mengakses Memori External 48
2.7.10 Sistem Interrupt 50
2.7.11 Pewaktu/Pencacah (Timer/Counter) 53
2.7.12 Port Serial 57
2.7.12.1 Baud Rate 58
2.7.12.2 Register Serial Port Control (SCON) 58

2.7.12.3 Serial Data Buffer Register 60
2.7.12.4 Program Status Word Register (PSW) 61
2.8 CODEBASE 62
2.8.1 Unsur CodeBase 62
2.8.2 Inisialisasi CodeBase 65
2.8.3 Open Dan Close Data File 65
2.8.4 Akses ke Field 66
2.8.4.1 Dengan Input Nomor Field 66
2.8.4.2 Dengan Nama Field 67

BAB III PERENCANAAN DAN PEMBUATAN 68
3.1 PERENCANAAN PERANGKAT KERAS 68
3.1.1 Unit Power Supply 68
3.1.2 Unit Modem 69
3.1.3 Unit Current Loop Comunication 70
3.2 PEMBUATAN PERANGKAT LUNAK 71
3.2.1 Software Untuk Serial COM 71
3.2.2 Untuk Akses Data Base File 75
3.2.2.1 Inisialisasi, Open, Dan Close Data File 75
3.2.2.2 Edit, Tambah, Dan Hapus Data 76
3.2.2.3 Search Record 78

BAB IV CARA PEMAKAIAN ALAT 79
4.1 Persiapan 79
4.2 Di Terminal Kontrol 79
4.3 Program Route.Exe di Server Kontrol 80
4.4 Program Svr.Exe di Server Kontrol 80

4.4.1 Penambahan Data Base 80
4.4.2 Penghapusan 81
4.4.3 Pengeditan..... 81
4.4.4 Melihat Report 81
4.4.5 Keluar (ESC) 82

BAB V PENUTUP..... 83
5.1 Kesimpulan 83
5.2 Saran 84

DAFTAR PUSTAKA
LAMPIRAN
USULAN TUGAS AKHIR
RIWAYAT HIDUP

DAFTAR GAMBAR

2-1 Bentuk Suatu Field 15

2-2 Susunan bit dalam Pengiriman Data Serial Asinkron..... 20

2-3 Transmisi Data Digital dgn Menggunakan Modem

dan Jalur Telpon Standar 26

2-4 Arsitektur Internal 8051 30

2-5 Konfigurasi Pin 8031 31

2-6 Diagram Blok Mikrokontroler 8031..... 34

2-7 Organisasi Memori 8031 41

2-8 Ruang Alamat Memori Data Internal 42

2-9 Alamat-alamat Bit RAM Internal 43

2-10 Alamat-alamat Register Fungsi Khusus 44

2-11 Struktur Port 0, 1, 2, dan 3 47

2-12 Siklus Waktu Pembacaan Memori Program 50

2-13 Siklus Waktu Pembacaan Memori Data 50

2-14 Siklus Waktu Penulisan Memori Data 51

2-15 Register Interrupt Enable 51

2-16 Register Interrupt Priority 52

2-17 Register Timer/Counter Control/Status TCON..... 54

2-18 Model Timer/Counter 56

2-19 Register Timer/Counter Mode (TMOD) 56

2-20 Register SCON 59

2-21 Register PSW 62

2-22 Struktur Codebase..... 63

3-1 Rangkaian Unit Power Supply 68

3-2 Blok Unit Prosessor 69

3-3 Rancangan Rangkaian Modem 70

3-5 Rancangan untuk IC 1488 71

3-6 Rancangan untuk IC 1489 71

DAFTAR TABEL

2-1 Tabel Tipe Data dalam Bahasa C 9

2-2 Tabel AT Command 22

2-3 Tabel Dial Command Modifier 23

2-4 Register Fungsi Khusus (SFR) Mikrokontroller 8031 37

2-5 Register-register Fungsi Khusus 45

2-6 Alamat Awal Program Pelayanan Interrupt 52

2-7 Nama dan Arti IE Register 52

2-8 Nama dan Arti Interrupt Priority Register 53

2-9 Nama dan Arti Register TCON 54

2-10 Pemilihan Mode Operasi Bit Serial 60

BAB I
PENDAHULUAN

1.1 Latar Belakang

Hal yang melatarbelakangi pembuatan tugas akhir ini adalah permasalahan yang sering dijumpai di pabrik-pabrik, yang menempati areal yang luas. Biasanya pabrik-pabrik besar tersebut jumlah karyawan yang dipunyai juga sangat banyak. Masalah ini sering berakibat berkurangnya pengawasan pada hal-hal yang bersifat sekuritas tinggi. Dengan kata lain dengan pabrik yang luas dan karyawan yang tidak sedikit berakibat makin bebasnya seorang karyawan, atau malah sama sekali bukan karyawan dari perusahaan itu, yang lalu lalang di dalam pabrik.

Dengan semakin banyaknya jumlah manusia, makin banyak pula kebutuhan yang harus dipenuhi. Sehingga makin banyak pula industri-industri bermunculan, mulai dari pengolahan makanan, kosmetik dan obat, sampai industri berat.

Di dalam sebuah pabrik, misalnya pabrik kimia, biasanya terdapat laboratorium atau ruangan khusus. Dimana hanya orang-orang tertentu yang diijinkan masuk ke laboratorium tersebut. Ini disebabkan dalam laboratorium itulah awal mula perancangan dan pembuatan obat atau bahan kimia khusus. Yang seringkali formula bahan kimia ini sengaja dirahasiakan, terutama terhadap pabrik obat lainnya, agar tidak ditiru. Dengan kata lain kunci utama dari pabrik itu terletak dalam laboratoriumnya.

Karena itulah dirasakan perlunya pengamanan khusus di dalam pabrik. Diantaranya bisa dengan dibuatkan pos penjagaan. Jadi tiap kali ada orang yang ingin melewati pintu tersebut, maka ia harus menunjukkan kartu identitasnya (atau biasa disebut ID Card) pada petugas. Setelah diijinkan barulah ia dapat masuk.

Cara lain adalah dengan pemasangan Tombol Password. Bila seorang karyawan ingin melewati pintu tersebut, ia dihadapkan pada serangkaian tombol-tombol. Dengan menekan kode tertentu, pintu akan terbuka.

Yang lain lagi ada yang menggunakan gabungan sistem sekuritas. Misalnya digunakan kamera monitor dan serangkaian tombol Password. Dimana kamera monitor itu pintu akan terus-menerus dipantau oleh petugas dari posnya. Hal ini berarti dibutuhkan petugas Satpam yang harus siaga 24 jam.

1.2 Permasalahan

Masalah itu masih mudah untuk diselesaikan bila dalam perusahaan itu tidak terlalu banyak jumlah karyawannya, atau paling tidak jumlah karyawan yang harus mengakses ke satu laboratorium tersebut. Tetapi seringkali karyawan yang harus bisa masuk kesana sangatlah banyak, bisa mulai dari direktur sampai tukang sapu.

Bila dengan menggunakan kartu identitas yang harus ditunjukkan pada Satpam, bisa saja si Satpam tidak melihat pada foto wajah yang terdapat pada kartu tersebut. Dan dengan tenangnya 'karyawan' itu langsung diijinkan masuk. Padahal mungkin saja orang yang dianggap karyawan oleh si Satpam itu baru kemarin

dipecat. Sedang ternyata kartu identitas yang dibawanya hanyalah sepotong kertas yang diberi warna spidol.

Belum lagi bila dilihat bahwa ternyata seorang Satpam profesional-pun tidak akan bisa 24 jam duduk di kursi terus-menerus menunggui pintu masuk. Yang sering adalah si Satpam lebih senang berkumpul di pos Satpam bersama-sama Satpam yang lainnya. Padahal, walaupun masih terletak di kompleks yang sama, jaraknya berjauhan dengan laboratorium khusus yang seharusnya dijaganya.

Bila yang dipakai adalah papan tombol kode, untuk kunci pintu tersebut, sekilas kelihatan aman. Tapi ketika seseorang karyawan yang akan masuk menekan angka-angka itu, ia sama sekali tidak memperhatikan apakah ada orang lain yang mengintip. Biarpun dengan konfigurasi tombol yang setiap beberapa waktu sekali diganti, biasanya dengan 2 kali melihat saja sudah dapat dihapal oleh orang asing itu.

1.3 Tujuan

Untuk itu terasa perlu adanya sebuah pengontrol pintu otomatis, dengan kunci pembuka yang unik untuk setiap karyawan yang boleh akses. Karena perlu pula jarak yang berjauhan, digabungkan teknik pengiriman data dengan Frequency Shift Keying (FSK). Yang mana keseluruhan sistem dijalankan dengan software, dengan mengutamakan basis data karyawan, dapat dikontrol karyawan yang keluar atau masuk.

1.4 Pembatasan Masalah

Tugas akhir ini adalah merancang dan membuat pengontrol pintu otomatis, dimana dengan data base yang ada akan dapat menentukan bisa tidaknya pintu tersebut dibuka. Selain itu akan dapat diketahui pula hari apa dan jam berapa seseorang mengakses pintu.

Compiler yang dipakai adalah Borland C++ versi 3.1, oleh Borland International Corp. Yang digabungkan dengan software untuk manajemen basis data modern, yaitu Code Base versi 5 yang dikeluarkan oleh Sequiter Software Inc.

Perangkat kerasnya dibagi dua jenis. Yang pertama adalah Terminal Contoller, dan yang kedua adalah Server Controlle. Pada Terminal Kontrol dibutuhkan konfigurasi minimal sebuah sistem minimum dengan Input Device (Keyboard), denagan sebuah Serial Port (RS-232-C). Sedang untuk Stasiun Server Kontrol, dibutuhkan konfigrasi minimal PC-286, dengan memori minimal 2 MB, harddisk, dan sebuah Serial Port (RS-232-C), dan juga monitor dengan card CGA standar.

1.5 Metodologi

Untuk mencapai tujuan yang telah disebutkan di atas dilakukan langkah-langkah sebagai berikut :

Langkah pertama adalah pengumpulan literatur-literatur sehubungan dengan masalah yang dihadapi. Kemudian dilanjutkan dengan pemahaman tentang IC Frekwensi Shift Keying (FSK), dan teknik pemakaiannya, standar RS-232-C

dengan teknik pemrogramannya dan sistem minimum dengan mikroprosessor 8031, beserta komponen-komponen elektronika penunjangnya.

Langkah selanjutnya adalah merencanakan sistem modem-nya dengan FSK dan software pendukungnya, yang sesuai dengan karakteristik operasionalnya. Perencanaan dilanjutkan dengan sistem perangkat lunak yang mendukung seluruh operasi alat dan sistem data base-nya.

Dari hasil rancangan tersebut, dibuat perangkat keras dan perangkat lunaknya. Baru kemudian dilanjutkan dengan pengujian alat. Dari hasil pengujian alat tersebut dapat diperoleh suatu kesimpulan dari keseluruhan sistem yang telah dirancang dan dibuat. Akhirnya, dari seluruh langkah-langkah di atas disusun dalam laporan Tugas Akhir ini.

1.6 Sistematika Penulisan

Sistematika dari laporan Tugas Akhir ini, yang terdiri dari 6 (enam) bab, disusun sebagai berikut :

Bab I adalah pendahuluan, yang berisi tentang latar belakang, permasalahan, tujuan, metodologi, sistematika penulisan dan relevansi.

Bab II adalah teori penunjang, yang membahas sistem minimum mikroprosessor 8031 beserta mode-mode yang terdapat dalam pengunaanya, sistem komunikasi RS- 232-C, tipe-tipe data dalam bahasa C, dasar-dasar basis data, modem, unsur-unsur Codebase seperti Function, Structure, dan Constant, cara inisialisasi, Open, Close, dan cara akses Field.

Bab III adalah perencanaan dan pembuatan, yang mencakup 2 macam yaitu, perencanaan perangkat kerasnya, dan pembuatan perangkat lunak. Perencanaan perangkat keras berisi tentang perancangan unit power supplai, unit modem dan perangkat penunjang lainnya. Sedang pembuatan perangkat lunak, yang meliputi perangkat lunak dari sistem di Terminal Controller, dan perangkat lunak di sistem Server Controller beserta sistem database-nya.

Bab IV adalah berisi tentang cara pemakaian peralatan, termasuk sistem databasenya. Seperti menambah dan mengurangi data.

Bab V adalah penutup yang berisi kesimpulan dan saran-saran, yang mana diketahui setelah peralatan yang dibuat sudah dicoba, yang memungkinkan pengembangannya.

1.7 Relevansi

Dari tugas akhir ini dapat diperoleh manfaat berupa penerapan dari ilmu-ilmu yang telah diperoleh selama kuliah di Jurusan Teknik Elektro, Bidang Studi Teknik Komputer, FTI ITS.

Hasil rancangan dan pembuatan alat pada tugas akhir ini dapat dimanfaatkan oleh pengguna dan penggemar elektronika dalam perancangan dan pembuatan rangkaian elektronika yang menggunakan teknik frekwensi shift keying (FSK).

Diharapkan dari hasil perancangan yang telah dilakukan, khususnya sistem pengontrolan dan data basenya yang menggunakan perangkat lunak, dapat digunakan sebagai referensi dalam perancanngan lainnya.

BAB II
TEORI PENUNJANG

2.1 Pengontrol Pintu Untuk Karyawan

Pada umumnya pintu-pintu masuk, semisal di pabrik-pabrik, atau di ruangan-ruangan perusahaan strategis dan militer, atau juga di hotel-hotel berbintang empat atau lebih, dan juga ATM (Automatic Teller Machine), dipakai pintu-pintu berkunci elektronik. Digunakannya kunci-kunci elektronik ini dengan tujuan agar ruangan- ruangan tersebut lebih aman. Tujuan lainnya adalah untuk pembatasan sehigga hanya orang-orang tertentu saja yang diperkenankan masuk atau melewati (Acces) pintu tersebut.

Kunci-kunci pintu ini beragam dalam jenisnya. Ada yang dengan kartu magnetik, ada yang dengan tipe Barcode, dan ada pula dengan digunakannya nomor-nomor pin yang harus diketikkan pada suatu keyboard.

Pada sistem seperti di hotel-hotel, digunakan kartu magnetik untuk membuka kunci elektronik pada pintu. Kartu dengan pita magnetik di tengah-tengah pada sisi belakang ini, harus diisi berulangkali dengan data tertentu untuk tiap pintu. Seringkali data di pita tersebut hilang, karena pita sering terkena tangan atau dekat dengan benda yang bermagnetik. Di hotel-hotel tersebut biasanya dipasang kunci elektronik yang *stand alone* (berdiri sendiri). Jadi kunci ini hanya sekali diprogram untuk satu kode saja.

Pada sistem yang lebih baik dipakai pada perusahaan-perusahaan dan industri-industri. Misalnya pada industri strategis. Disini kunci elektronik tidak dipasang *standalone*, tetapi tiap-tiap kunci elektronik tersebut dihubungkan dengan suatu pengontrol pusat. Dimana biasanya sistem ini dihubungkan juga dengan sistem peralatan kamera monitor. Jadi memang sistem yang ini seperti sistem sekuriti.

Sistem yang mirip seperti diatas ada pada ATM. Dimana tiap ATM dihubungkan dengan sistem pusat. Bila ada pelanggan yang akan mengambil uang, data pelanggan dari kartu magnetik dikirimkan lewat saluran telepon, ke sistem pusat. Disini data dibandingkan dengan database yang ada di mini/mainframe. Bila si pelanggan punya account yang cukup, ATM akan diperintahkan oleh sistem pusat tersebut untuk memberikan jumlah uang yang diminta.

Pada Pengontrol Pintu untuk Karyawan ini dipakai sistem seperti ATM di atas. Yang pertama, untuk itu diperlukan sistem pada bagian penerima datanya. Pada sistem ini cukup dipergunakan sistem yang minim. Karena setelah data dibaca, dan kemudian dikirimkan, sistem tinggal menunggu perintah selanjutnya dari pusat pengontrol.

Yang kedua sistem Database di pusat pengontrol adalah bagian yang sangat penting, karena banyaknya jumlah karyawan. Karena kecepatan pengolahan data karyawan akan menjadi kritis bila jumlahnya terus bertambah.

Dan terakhir yang sangat penting pula adalah sistem pengiriman data. Karena jarak yang berjauhan antara pintu dan pusat pengontrol, seperti di pabrik

petrokimia, data dikirimkan lewat saluran telpon, biasanya dipakai MODEM. Sebab pada umumnya pada tiap pabrik sudah tersedia jaringan telpon pada tiap bangunannya. Kecepatan seluruh sistem akan ditentukan oleh kecepatan sistem pengiriman data oleh bagian Modem, sehingga seluruh sistem akan lebih efisien bila dipakai Modem yang berkecepatan tinggi.

2.2 Struktur Data

Komputer merupakan suatu alat yang dapat digunakan untuk mengolah suatu informasi. Sedangkan ilmu komputer sendiri mempelajari tentang bagaimana suatu data dapat dikelompokkan dan diorganisasikan, ataupun diolah untuk kemudian didapat hasil berupa informasi yang bermanfaat.

Teori tentang struktur data mencakup teori tentang pengorganisasian data yang diolah dan teori tentang metoda pengolahan data. Pengetahuan yang baik tentang struktur data ini sangat berguna dalam pembuatan suatu perangkat lunak. Pengorganisasian data yang baik diantaranya akan menjamin penghematan ruang penyimpanan maupun kemudahan dalam mengolah data itu sendiri. Sedangkan pemilihan metoda yang tepat untuk mengolah data memungkinkan proses berjalan efisien dan cepat.

2.3 Tipe Data

Data dalam bahasa pemrograman adalah suatu nilai yang dapat dinyatakan dalam bentuk 'konstanta' atau *variabel.* Konstanta menyatakan nilai yang tetap,

sedangkan variabel menyatakan nilai yang berubah-ubah selama eksekusi berlangsung.

Uraian berikut akan membahas tipe data dalam bahasa pemrograman C yang digunakan dalam tugas akhir ini.

2.3.1 Tipe Data Dasar

Tipe data dasar dalam bahasa pemrograman C, berdasarkan jenisnya dapat dibagi dalam kelompok-kelompok dibawah ini :

1. Bil. Bulat (integer) dengan kata kunci *int*.
2. Bil. Pecahan (real) presisi tunggal dengan kata kunci *float*.
3. Bil. Pecahan (real) presisi ganda dengan kata kunci *double*.
4. Karakter dengan kata kunci *char*.
5. Data tak bertipe dengan kata kunci *void*

Disamping tipe-tipe data diatas masih adalagi pemodifikasian tipe (type modifier) yang dapat dikenakan pada awal tipe data dasar, diantaranya *signed* dan *unsigned*. Total *bit* yang dibutuhkan untuk menyimpan data dan *range* dari tipe data dapat digambarkan pada tabel 2.1.

Tabel 2.1[1] : Tipe Data dlm C

Tipe Data	Total Bit	Range
unsigned char	8	0 s/d 255

Tabel berlanjut ke halaman berikut.

[1] Borland, Borland C++ 3.1 Programmer Guide, Borland International, hal. 19

Lanjutan Tabel : Tipe Data dlm C

Tipe Data	Total Bit	Range
char	8	-128 s/d 127
enum	16	-32768 s/d 32767
unsigned int	16	0 s/d 65535
short int	16	-32768 s/d 32767
int	16	-32768 s/d 32767
unsigned longf	32	0 s/d 4294967295
long	32	-2147483648 s/d 2147483647
float	32	3.4 *(10**-38) s/d 3.4 * (10**+38)
double	64	1.7 * (10**-308) s/d 1.7 * (10**+308)
long double	80	3.4 * (10**-4932) s/d 1.1 * (10**+4932)

2.3.2 Array

Array atau larik merupakan koleksi data yang mempunyai tipe sama. Setiap elemen dari 'array' menggunakan nama yang sama dan hanya dibedakan memlalui index atau urutannya.[2]

Dalam bahasa C, besar array adalah tetap dan disimpan dalam memori yang berurutan.

---

[2] Ibid., hal. 245

2.3.3 Structure

Istilah *structure* yang dikenal dalam bahasa pemrograman C, dalam bahasa lain dikenal sebagai *record.*

Secara umum 'record' merupakan kumpulan informasi mengenai sebuah kesatuan. Sedangkan dalam bahasa pemrograman C, sebuah 'record' (structure) adalah koleksi dari variabel yang dinyatakan dengan sebuah nama. Variabel-variabel ini dapat mempunyai tipe yang berbeda.[3]

2.3.4 Linked List

List adalah kumpulan dari obyek atau elemen (data) yang tersusun menurut aturan tertentu.[4] Elemen-elemen dalam 'list' tidak terurut akan tetapi tetap dapat diadakan penambahan, perubahan ataupun penghapusan elemen dalam 'list' karena posisi dari elemen dapat diketahui.

Adapun sifat-sifat dari 'list' adalah sebagai berikut:

1. sebuah 'list' dapat terdiri dari 0 buah elemen atau lebih.

2. elemen baru dapat ditambahkan ke dalam 'list' pada posisi yang diinginkan.

3. tiap elemen dalam 'list' dapat dihapus

4. tiap elemen dalam 'list' dapat diakses

---

[3] Ibid., hal. 65

[4] Grady, M. Tim, C! Programming Principles & Practices, McGraw-Hill, 1989, hal. 215

Linked list adalah 'list' yang terdiri dari beberapa set *node* (titik), dimana tiap 'node' minimal terdiri dari dua *field*, satu 'field' untuk menunjukkan data, dan 'field' lainnya menunjuk ke *node* selanjutnya di dalam 'list'.

Setiap 'node' dari 'linked list' dapat dialokasikan pada memori pada saat dibutuhkan dan dibebaskan pada saat tidak dibutuhkan lagi, jadi 'linked list' ini bersifat fleksibel. Ini disebabkan karena cara penyimpanan 'node' pada 'linked list' tidak selalu harus berurutan seperti 'array'.

Proses-proses yang umum terjadi pada 'linked list' adalah pencarian data, penambahan, dan penghapusan 'node'.

2.3.5 File

Satuan informasi terkecil adalah 'bit' yang bernilai 1 atau 0. 'Bit-bit' ini membentuk satuan informasi yang lebih besar yang disebut karakter atau *byte*. Karakter-karakter ini menyusun *field*, dan 'field-field' akan membentuk suatu *record*. Dan terakhir, record-record inilah yang akan membentuk *file*.

2.4 Basis Data

Konsep Basis Data gampangnya adalah seperti sebuah kumpulan informasi yang diorganisasi dan disusun dalam susunan tertentu.[5] Semisal yang disusun dalam basis data adalah buku telepon. Di dalamnya berisi nama-nama, nomor-nomor telpon, dan alamat-alamat dari ratusan orang. Tiap daftar di buku telpon berarti

---

[5] Sequiter, CodeBase 5.0 User's Guide, Sequiter Software Inc., hal. 19

sama dengan satu 'Record', dan tiap butir infomasi dalam Record juga berarti sama dengan satu 'Field'.

Seperti ditunjukkan di bagian bawah tiap 'Data File' (biasanya disebut sebuah tabel) adalah kumpulan satu atau lebih Field (biasa disebut sebuah *Tuple*). Tiap Field mempunyai satuan khusus yang menunjukkan ukuran dan jenis data yang akan disimpan.

Secara keseluruhan deskripsi field-field ini membentuk strukture dari sebuah Data File. Database adalah sebuah grup organisasi informasi yang bisa dibagi-bagi diantara beberapa Data File yang berbeda. Data File adalah sebuah file yang menyimpan informasi dalam sebuah bentuk terorganisir. Umum Data File mempunyai sebuah file dengan ekstensi ".DBF". Field berarti bagian terkecil dari informasi yang terkandung dalam sebuah Record. Record adalah sebuah kumpulan dari beberapa Field dalam Data File.

Bentuk susunan buku telpon digambarkan pada Gambar 2.1.

Field Nama	Field Alamat	Field Telpon #
Budi	Perum. AL Blok H-2	3291-723
Cahyo	Perum ITS P-3 Keputih	
Gan	Perum ITS P-3 Keputih	
Joko Santoso, Ir	Perum ITS J-2 Keputih	5946-500
Kusnan	Jl. Gubeng 41	
Zainnudin	Jl. Kertajaya Indah 5	5239-230

Gambar 2.1 : Bentuk Suatu Field

2.4.1 Field Dan Record

DBF standar file mempunyai 4 atribut untuk mendefinisikan tiap filed-nya. Atribut ini disebut 'Nama', 'Tipe', Panjang' dan 'Desimal'.[6]

1. Nama: Ini sesuai dengan sebutannya adalah untuk nama dan mengidentifikasikan field-nya. Tiap nama bisa mempunyai maximum 10 karakter, dan masing-masing field nama harus unik (Unique) bila terdapat dalam file data yang sama. Dan juga harus mengandung alfanumerik atau karakter biasa.

2. Tipe: Tipe dari field menentukan informasi apa yang harus disimpan disana. Terdapat 6 jenis tipe yang berbeda yang bisa didefinisikan untuk tiap field-nya. Yaitu karakter, numeric, Floating Point, tanggal, Logical dan memo.

3. Panjang: Atribut ini berisi jumlah dari karakter atau digit yang bisa disimpan dalam satu field-nya.

4. Desimal: atribut ini hanya berlaku untuk field numerik atau Floating Point. Ini menentukan jumlah digit setelah angka desimal.

Sebuah record berisi satu baris panjang yang terbentuk dari seluruh field, dan mempunyai nomor record dan sebuah *Deletion Flag* yang unik dengan yang setaraf dengannya. Nomer record berisi tetang posisi fisik dalam file data, dimana ini diperlukan oleh Deletion Flag untuk proses penghapusan record-nya. Deletion Flag di set True untuk mengindikasikan bahwa record itu harus di hilangkan dari file data ketika file data di-pak (packed).

---

[6] Ibid., hal. 20

Deletion Flag mengindikasikan benar atau tidaknya sebuah record sudah ditandai untuk dihapus. Nomer record adalah sebuah nomer yang unik, yang menyatakan posisi fisik dari sebuah record dalam data file.

2.4.2 Tag, Index Dan Filter

Sebuah *Tag* yang mana berisi perintah dengan cara bagaimana record-record dalam file data tersebut dikeluarkan.[7] Perintah dalam Tag tidak mempengaruhi langsung secara fisik pada record di file data, hanya perintah bagaimana caranya agar record tersebut dapat di-akses. Informasi yang dikandung perintah ini disebut kunci index (Index Key).

Sebuah index adalah sebuah file yang mengandung sorting kunci-kunci index untuk satu atau lebih Tag. Terdapat beberapa format file index yang biasa dipakai :

1. ".NDX" : pada dBASE III, dan dBASE III PLUS
2. ".MDX" : pada dBASE IV
3. ".NTX" : pada Clipper
4. ".CDX" : pada FoxPro

".CDX" dan ".MDX" membolehkan untuk mempunyai tag dalam tiap file index, dan dapat mempunyai *Production Index*. Sebuah produk index adalah sebuah file index yang dibuka otomatis ketika sebuah data file dibuka. ".NDX" dan ".NTX" format akan membatasi hanya bisa satu buah tag-nya pada tiap file index-nya. Dan juga tidak diperbolehkan untuk memproduk index.

---

[7] Ibid., hal. 21

Index adalah sebuah file yang berisi satu atau lebih tag. Produk index adalah sebuah file index yang secara otomatis dibuka ketika sebuah data file dibuka. Kunci index adalah satu bagian dari sebuah record yang dipakai oleh sebuah perintah tag sebagai dasarnya. Tag berisikan perintah dengan cara bagaimana record di file data harus dikeluarkan.

2.5. KOMUNIKASI DATA

Komunikasi didalam sistem komputer umumnya dilakukan secara paralel lewat saluran data bus. Yaitu bisa berupa word atau byte. Disamping untuk keperluan intern, diperlukan juga transfer data antar komputer, baik untuk jarak lokal maupun berjauhan. Komunikasi data terbagi dalam 2 (dua) bagian, yaitu :

1. Komunikasi data paralel
2. Komunikasi data serial

2.5.1. KOMUNIKASI DATA PARALEL

Komunikasi data paralel dilakukan dengan cara pengiriman atau penerimaan 8 bit data sekaligus pada waktu yang sama. Perpindahan data dilakukan byte demi byte atau word demi word sehingga memiliki kecepatan relatif tinggi.

Komunikasi data paralel membutuhkan jumlah kawat saluran sejumlah bit yang diperlukan. Oleh karena itu pemindahan data secara paralel ini tidak sesuai

untuk komunikasi jarak jauh. Disamping itu untuk jarak jauh akan menimbulkan cross talk derau yang cukup besar.

2.5.2. KOMUNIKASI DATA SERIAL

Komunikasi Data Serial terkadang suatu hal yang sulit dipahami, karena membutuhkan bantuan beberapa topik yang lain untuk bisa memahaminya.

Dalam mikrokomputer data ditransfer dalam bentuk paralel, karena cara inilah yang paling cepat untuk dapat menyelesaikan[8]. Tapi untuk men-Transfer data dengan jarak yang cukup jauh, Transmisi data paralel memerlukan kabel yang banyak. Karenanya untuk mengirim data melalui jarak yang jauh, biasanya data paralel tersebut dirubah dulu menjadi bentuk serial, sehingga hanya membutuhkan seutas kawat atau bisa dengan kabel ganda. Setelah data serial yang diterima dari tempat lain yang jauh, hasilnya diubah kembali menjadi bentuk paralel, sehingga dengan mudahnya dapat ditransfer ke Bus Microcomputer. Tiga bentuk sistem pengiriman data serial : Simplex, Half-Duplex, dan Full Duplex.

1. Simplex : Jalur data Simplex hanya bisa mengirim data satu arah saja.

2. Half-Duplex : Transmisi Half-Duplex berarti komunikasi dapat terjadi dua arah diantara dua perangkat tersebut, tetapi hanya terjadi satu arah saja pada satu waktu. Contoh: Radio komunikasi.

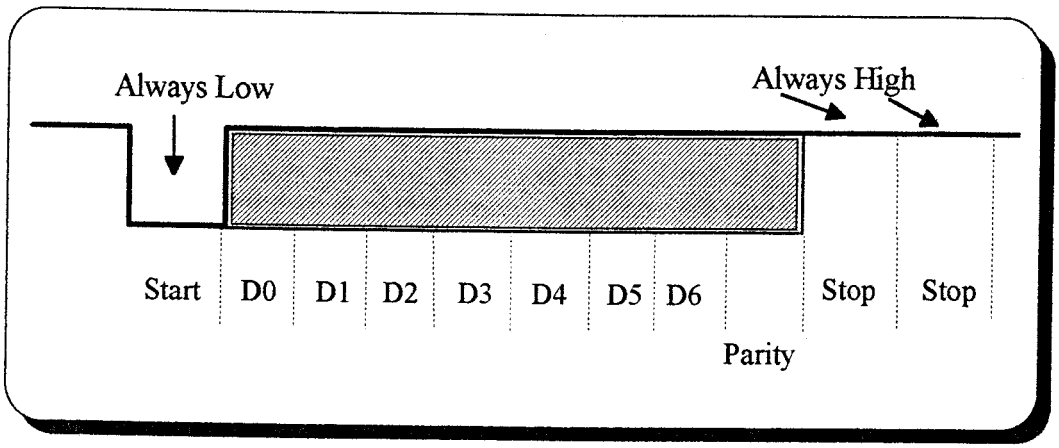
3. Full Duplex : Artinya bahwa tiap sistem atau perangkat tersebut dapat mengirim dan menerima data pada saat yang sama. Contoh: Telepon.

---

[8] Douglas V. Hall, Microprosessor And Interfacing: Programming & Hardware, McGraw-Hill, 1992, hal. 488

Serial Data dapat dikirim dalam bentuk Sinkron atau Asinkron. Untuk transmisi sinkron, data dikirim dalam blok-blok dengan jangka waktu tetap. Start dan akhir tiap blok ditandai dengan byte-byte atau bit-bit tertentu.

Pada transmisi asinkron tiap karakter data, diberi sebuah bit untuk start, dan 1 atau 2 buah bit sebagai tanda akhir. Karena tiap karakter diidentifikasikan sendiri-sendiri, karakter dapat dikirim kapanpun (asinkron), seperti bila seseorang mengetik di Keyboard.



Gmb 2-2 : Susunan bit dalam pengiriman data serial asinkron[9].

Gambar 2-2 menunjukkan susunan bit pada saat mengirim data serial dengan transmisi asinkron. Ketika tak ada data yang dikirim, sinyal akan memberikan kondisi High dengan konstan, atau dalam kondisi *marking*. Mulainya karakter data ditandai dengan menjadi Low-nya 1 bit waktu. Bit ini disebut Start

---

[9] Douglas V. Hall, Microprosessor And Interfacing: Programming & Hardware, McGraw-Hill, 1992, hal. 488

pertama kali. Sehingga satu Data Word akan menampung 5, 6, 7, atau 8 bit. Setelah bit data diikuuti bit Parity, dimana digunakan untuk men-cek eror setelah data diterima. Beberapa sistem tidak memasukkan atau melihat bit Parity. Setelah mengirim bit data dan bit Parity, sinyal akan dikembalikan ke posisi High minimal selama jangka waktu 1 bit waktu untuk menandai akhir dari pengiriman karakter tersebut. Bit yang selalu-High ini sebagai tanda *Stop Bit*. Pada sistem-sistem lama digunakan 2 Stop bit.

2.6 Modem

2.6.1 Dasar Modem

Sinyal digital, semisal teleprinter, tidak dapat langsung dikirimkan melalui jalur telpon. Karena jalur telpon itu hanya bisa dilewati oleh sinyal suara biasa. Jadi range-nya terbatas hanya berada di *bandwith* 200 s/d 8000 Hz.[9] (Suara manusia pada umumnya berkisar antara 300 s/d 3000 Hz.). Untuk itulah diperlukan Modulator yang merubah sinyal digital menjadi sinyal analog, dengan output frekwensinya berada pada range tersebut di atas. Untuk mengembalikannya diperlukan satu alat lagi yang disebut Demodulator.

Beberapa perintah dasar Modem (Modulator - Demodulator) :

---

[9] Campbell, Joe, ***C Programmer's Guide to Serial Communications***, Sams Publishing, 1994, hal. 136

Tabel 2.2 : AT Command; Seting Default dicetak **tebal**.

AT COMMANDS

CMD	KETERANGAN / DESCRIPTION
+++	Escape Code. Mengembalikan modem ke kondisi Command. Harus dilakukan delay 1 detik setelah dan sebelum kode ini.
AT	Attention Code. 'Precedes' seluruh Command kecuali A/ (Repeat Last Command) dan kode +++ Escape Code.
A	Answerd Phone Line. (This command is neither preceded by AT nor followed by a carriage return.)
B	Pergunakan protokol CCITT V.21/V.22
B1	Pergunakan protokol Bell 103/212A.
E*n*	Command echo (0=Disable, **1=Enable**).
I*n*	Return product ID or checksum information (0-5).
O	Return Online.
O1	Return online; initiate retrain sequence if at 2400 bps.
W*n*	Error correction call progress (**0=not reported and CONNECT XXXX reports DTE speed**, 1=reported, 2=not reported and CONNECT XXXX reports DCE speed).
X	Blind dial (ignore dial tone and busy signal);
&D	Ignore DTR signal.
&D1	Switch to asynchronous command state upon ON-to-OFF transitin of DTR.
&D2	Disconnect; return to command state. Auto answer disabled while DTR is OFF.

Tabel berlanjut ke halaman berikut.

Lanjutan Tabel : AT Command

&D3	Perform soft reset upon ON-to-OFF transition to DTR.
&K3	Enabled bidirectional hardware (RTS/CTS) flow control.
&K4	Enable software (XON/XOFF) flow control.
&K5	Enable transparent software (XON/XOFF) flow control.
&L*n*	Line selection (**0=dial up**, 1=not supported).
&M*n*	Same as &Q.
&P*n*	Pulse dial make/break ratio (**0=39/61 10pps**,
&Q	Mode selection (0=Direct async, 1=sync1, 2=sync2, 3=sync3, 5=error correction async, 6= Normal async).
&R*n*	RTS/CTS sync mode (**0=CTS responds to RTS**, 1=CTS on).
&S*n*	Select DSR action (0=always on, 1= EIA recommendation).
*H1	Negotiated link at 1200 bps.

Tabel 2-3 : Dial Command Modifier; Default dicetak **tebal**.

DIAL COMMAND MODIFIERS

MOD	DESCRIPTION
L	Dial last string dialed.
P	Pulse Dial.

Tabel berlanjut ke halaman berikut.

Lanjutan Tabel : Dial Command Modifier

R	Originate call is answer mode. Must be last character in dial string.
T	Tone dial.
W	Wait for dial tone before dialing. Pause before dialing next digit.
;	Return to command state after dialing.
!	Flash (go on hook for 700 ms.).
@	Wait for quit answer.
^	Turn on calling tone if originating call. Seting Default

2.6.2 Baud Rate

Baud-Rate digunakan untuk mengindikasikan pada kecepatan berapa data serial dikirimkan. Baud rate didefinisikan sebagai 1/(waktu transisi yang dibutuhkan sinyal). Bila sinyal berubah tiap 3.33 ms, baud ratenya adalah 1/(3.33 ms), atau 300 Bd. Kelihatannya mirip bila dianggap sebagai 300 bits/s. Tetapi pada banyak kasus 2 atau lebih data bit aktual dikodekan dalam satu sinyal transisi, sehingga data bit per detik dan baud sama sekali tidak berhubungan. Baud rate yang umum dipakai adalah 300, 600, 1200, 2400, 4800, 9600, dan 19200.

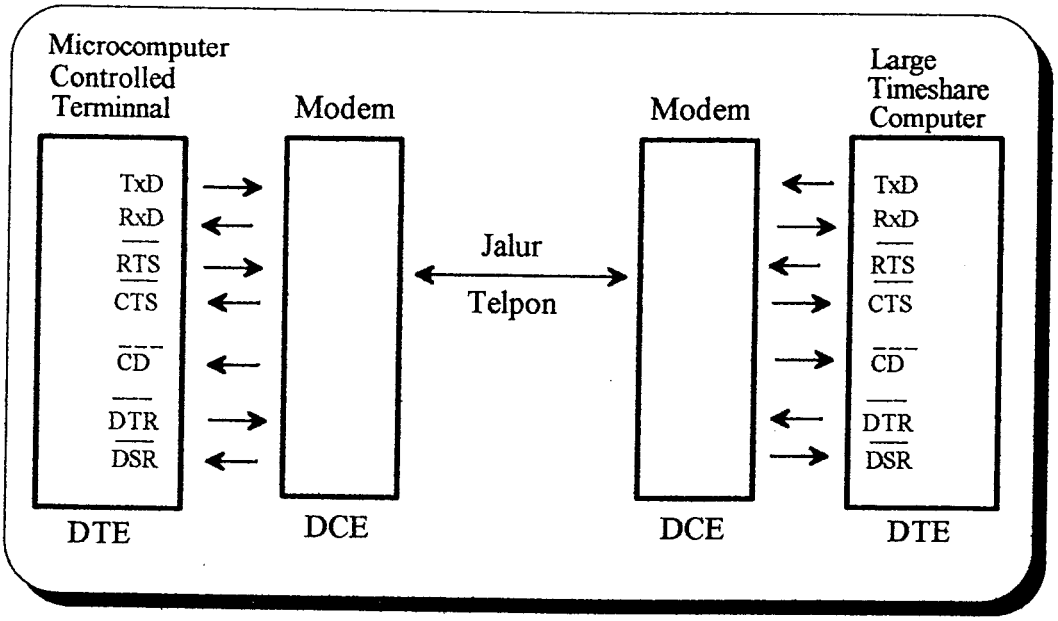
Untuk di-interface-kan ke microcomputer dengan memakai jalur data-serial, data harus diubah dari dan ke bentuk data-serial. Untuk ini sebuah shift register

Untuk di-interface-kan ke microcomputer dengan memakai jalur data-serial, data harus diubah dari dan ke bentuk data-serial. Untuk ini sebuah shift register *parallel-in-serial-out* dan sebuah shift register *serial-in-parallel-out* dibutuhkan. Selain itu diperlukan pula rangkaian *handshaking*, sehingga dapat dipastikan sistem pengirim tidak akan lebih cepat dalam pengiriman datanya daripada kemampuan sistem penerimanya.

Sekali data diubah menjadi bentuk serial, bagaimanapun juga harus dikirim dengan cara UART (Universal Asynchronous Receiver-Transmitter) dan diterima juga dengan UART.

Untuk jarak yang sangat jauh, data serial dikirim lewat sistem telepon standard. Ini menguntungkan karena pengkabelan dan penyambungan sudah ada di masing-masing tempat. Jalur telpon standard, bisa disebut juga *switched lines* karena dua titik manapun dapat dihubungkan melalui sebuah rentetan panjang saklar-saklar (Switch), hanya punya lebar-jalur (Bandwidth) antara 300 - 3000 Hz. Karena itu sinyal digital tidak dapat langsung dikirimkan lewat telpon standard. Gambar sinyal digital ini ditunjukkan di Gambar 2-2.

Solusinya adalah sinyal digital tersebut dirubah jadi suara audio (Audio-Frequency Tones), dimana suara itu masih dalam bandwidth yang bisa dikirimkan lewat jalur telpon. Peralatan untuk konversi ini dan juga untuk konversi kembali dari suara menjadi informasi digital disebut *Modem* (Modulator-Demodulator).



Gambar 2-3 : Transmisi data digital dengan menggunakan Modem dan jalur telpon standar.

Di Gambar 2-3 ditunjukkan bagaimana sebuah *Remote Terminal* berkomunikasi dengan sebuah komputer Mainframe[10], dengan bantuan 2 modem melalui jalur telpon. Modems dan peralatan lain yang digunakan untuk mengirim serial data lewat jarak yang sangat jauh disebut sebagai *Data Communication Equipment* (DCE). Terminal dan komputer yang mengirim atau menerima serial data disebut *Data Terminal Equipment* (DTE).

Nama data dan sinyal Handshake yang ditunjukkan Gambar 2-3 adalah bagian dari standard komunikasi data serial disebut RS-232C. Setelah power terminal dihidupkan dan terminal siap, ia dikirimkan sinyal *Data-Terminal-Ready* (DTR) untuk tanda bahwa modem siap. Karena modem sudah dihidupkan dan siap untuk mengirim atau menerima data, dijawab oleh modem dengan sinyal

---

[10] Ibid., hal. 489

Data-Set-Ready (DSR) ke terminal. Dibawah kontrol manual atau terminal kontrol, kemudian komputer di-dial oleh modem.

Bila komputer siap, dikirimkan kembali suara-suara khusus sebagai jawaban. Ketika terminal mempunyai sebuah karakter yang siap untuk dikirim, terminal akan memberi sinyal *Request-To-Send* (RTS) ke modem. Akan dijawab modem dengan mengirim balik sinyal *Carrier-Detect* (CD) yang artinya kontak ke komputer telah terhubungi. Ketika modem benar-benar sudah siap untuk mengirim data, sebagai tandanya dikirimkan kembali sinyal Clear-To-Send (CTS) ke Terminal. Kemudian diberikan oleh terminal data serial karakter yang ingin dikirim ke modem. Ketika terminal memberikan seluruh data serial karakternya, terminal harus membuat sinyal RTS dalam keadaan High. Ini membuat modem mengabaikan sinyal CTS dan menghentikan transmit-nya. Sehingga tampak ada handshake antara modem dan komputernya pada jalur data di ujung satunya.

2.7. MIKROKONTROLER 8031

Mikrokontroler sering disebut juga sebagai *Single chip microkomputer* karena dapat digunakan langsung sebagai unit pengontrol tanpa memerlukan bantuan komponen digital yang lain. Pada umumnya mikrokontroler digunakan untuk aplikasi-aplikasi khusus yang sudah diprogram dari pabrik. Banyak jenis mikrokontroler yang diproduksi oleh pabrik-pabrik komponen elektronik. Salah satu mikrokontoler yang banyak digunakan adalah keluarga dari MCS-51.

Mikrokontroler 8031 adalah salah satu mikrokontroler dari keluarga MCS-51. Selain 8031, anggota keluarga MCS-51 lainnya adalah 8051 dan 8751. MCS-51 dikemas dalam kemasan standar DIL (*dual in line*) 40 pin dan masing-masing mempunyai konfigurasi pin, pewaktu (*timing*) dan karakteristik listrik yang sama. Perbedaan utamanya adalah dalam hal memory program internalnya. Mikrokontroler 8751 mempunyai 4 kilo byte EPROM (*erasable programable read only memory*). Mikrokontroler 8051 mempunyai 4 kilo byte ROM (*read only memory*) yang telah diisi program (sesuai dengan kehendak pemakai) pada waktu proses pembuatan IC tersebut dan program tersebut tidak dapat diubah-ubah atau dihapus.Mikrokontroler 8031 tidak mempunyai memory program dalam dan hanya dapat menggunakan memory program eksternal.

MCS-51 dapat mengakses 64 kilo byte memory program eksternal dan 64 kilo byte memory data ekternal, mempunyai 32 jalur I/O dan sebuah *receive buffered,* serial I/O dua arah. Hal ini berarti MCS-51 dapat menerima byte yang kedua sebelum byte yang diterima belum dibaca dari receive register dan MCS-51 ini dapat mengirim dan menerima secara bersamaan.

2.7.1. ARSITEKTUR DAN ORGANISASI 8031

Mikrokontroler 8031 terdiri dari sebuah CPU (*centel procesing unit*), dua jenis memori yaitu memori data dan program, *port input/output,* dan register-register mode, status, dan data serta logika random yang diperlukan oleh berbagai fungsi periperal. Masing-masing bagian ini berhubungan satu dengan yang

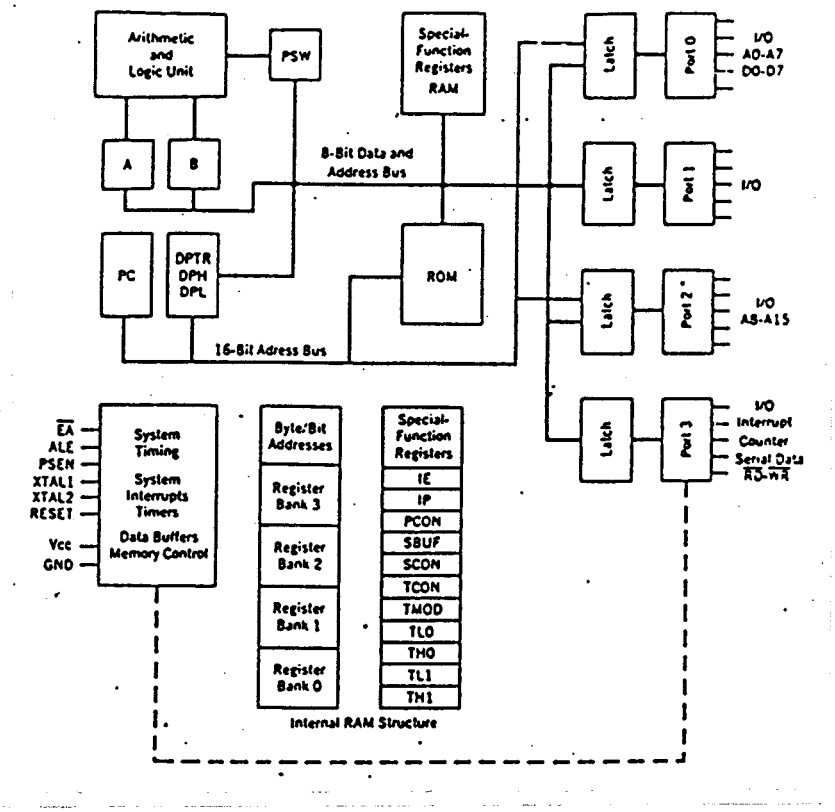
lain melalui data bus delapan bit. Bus ini dibuffer melalui port I/O bila diperlukan perluasan memori atau sebagai peralatan I/O.

CPU 8031 mempunyai empat ruang memori :

- 64 kilo byte memori program
- 64 kilo byte memori data eksternal
- 384 byte memori data internal
- 16 bit program counter

Alamat memori data internal selanjutnya dibagi dalam 256 byte RAM data internal dan 128 byte alamat register fungsi khusus (SFR). Empat register bank (masing-masing mempunyai delapan register), 128 bit yang dapat dialamati dan *stack* yang terdapat dalam RAM data internal.

Banyaknya stack dibatasi oleh kemampuan RAM data internal. Lokasinya ditentukan lewat 8 bit *stack pointer*. Semua register kecuali *program counter* dan empat *register banks* berada dalam ruang alamat (*address space*) register fungsi khusus (SFR). Register lain yang termasuk dalam SFR adalah register aritmatika, pointers, port I/O, dan register-register untuk *interupt, timer* dan *serial channel.* Lokasi-lokasi 128 bit dalam alamat SFR dapat dialamati sebagai bit-bit. Secara keseluruhan 8031 mempunyai 128 byte RAM data internal dan 20 register fungsi khusus.



Gambar 2.4 : Arsitektur Internal 8051[11]

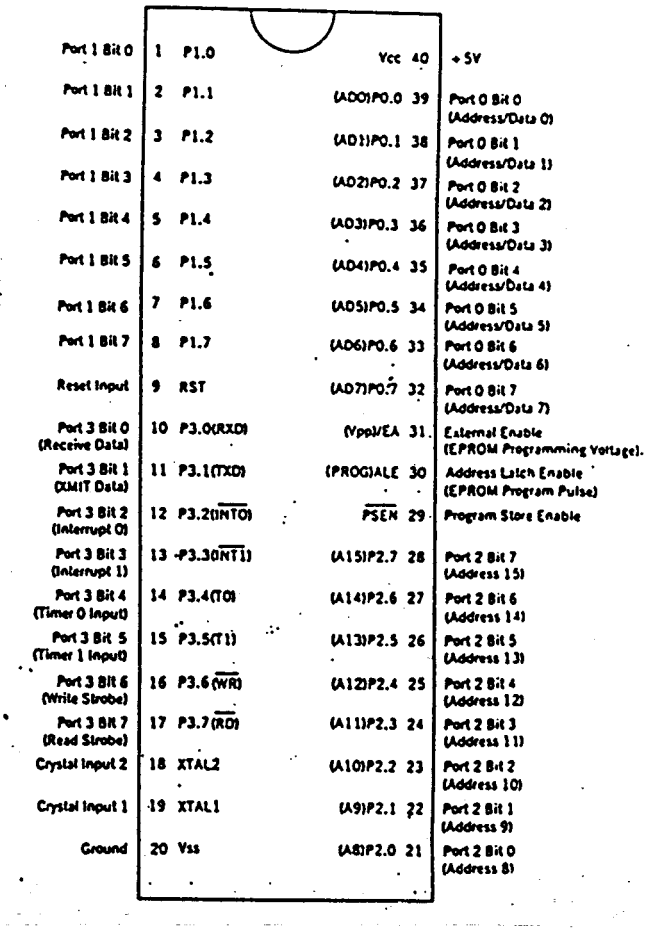
Instruksi-instrksi 8031 terdiri dari 49 buah *single byte,* 45 buah *two byte* dan 17 buah *three byte.* Bila digunakan kristal 12 Mhz sebagai osilatornya, 64 instruksi memerlukan waktu eksekusi 1uSc dan 45 instruksi mempunyai waktu eksekusi 2 *uSc.* Sedangkan 2 instruksi yaitu perkalian dan pembagian memerlukan waktu instruksi 4 *uSc.*

2.7.2. FUNGSI PIN-PIN 8031

Pada gambar 2.5 memperlihatkan konfigurasi pin-pin pada mikrokontroler 8031. Fungsi pin-pin pada 8031 dapat dikelompokkan menjadi pin sumber tegangan, pin kristal, pin kontrol, pin input-output dan pin interupsi.

---

[11] The 8051 Microcontroler,Architecture, Programing and Aplication , J. Ayala , p. 11



Gambar 2.5 : Konfigurasi Pin 8031[12]

Fungsi dari tiap pin tersebut adalah :

- Vcc :

Pin positip sumber tegangan 5 volt DC.

- Vss :

Pin ground sumber tegangan.

- Port 0 :

Port 0 merupakan port Input / Output delapan bit dua arah. Port ini dapat digunakan sebagai multipleks bus kealamat rendah dan bus data selama adanya akses ke memori program eksternal atau ke memori data eksternal.

---

12 Ibid, p. 15

- Port 1 :

Port 1 merupakan port Input / Output delapan bit dua arah. Setiap pin dapat digunakan sebai masukan atau keluaran tanpa tergantung dari pin lainnya.

- Port 2 :

Port 2 merupakan port Input / Output delapan bit dua arah ini dapat digunakan sebagai bus alamat tinggi selama adanya akses ke memori program eksternal atau memeori data eksternal.

- Port 3 :

Port 3 merupakan port Input / Output delapan bit dua arah. Port ini dapat digunakan juga untuk fungsi sebagai pin-pin istimewa bagi 8031, seperti sebagai berikut :

- P3.0 (RxD) : masukkan penerima data serial

- P3.1 (TxD) : keluaran pengiriman data serial

- P3.2 (INTO) : interupsi 0 ekternal

- P3.3 (INTI) : interupsi 1 eksternal

- P3.4 (TO) : masukkan ekternal pewaktu/pencacah 0

- P3.5 (TI) : masukkan ekternal pewaktu/pencacah 1

- P3.6 (WR) : strobe penulisan memori data eksternal

- P3.7 (WR) : strobe pembacaan memori data eksternal

- RST/VPD :

Pin ini berfungsi untuk mereset sistem 8031. Perubahan taraf tegangan dari rendah ke tinggi akan me-*reset* mikrokontroler.

- ALE/PRG :

ALE (*address latch enable*) digunakan untuk mengunci alamat rendah pada saat pengaksesan memori program eksternal selama operasi normal.

- PSEN :

PSEN (*program store enable*) adalah sinyal kendali yang menghubungkan memori program eksternal dengan bus selama operasi normal.

- EA/VDD :

Untuk pengoperasiain 8031, pin ini harus dihubungkan dengan ground agar dapat menjalankan instruksi dari program eksternal.

- XTAL 1 :

Pin ini merupakan masukkan ke penguat osilator berpenguat tinggi. Pin ini dihubungkan dengan kristal atau sumber osilator dari luar.

2.7.3 PERANGKAT KERAS UNIT PEMROSES PUSAT (CPU)

Pada gambar 2.6 memperlihatkan diagarm blok dari mikrokontroler 8031. Fungsi pin-pin pada 8031 dapat dikelompokkan menjadi pin sumber tegangan, pin kristal, pin kontrol, pin *input-output* dan pin interupsi.

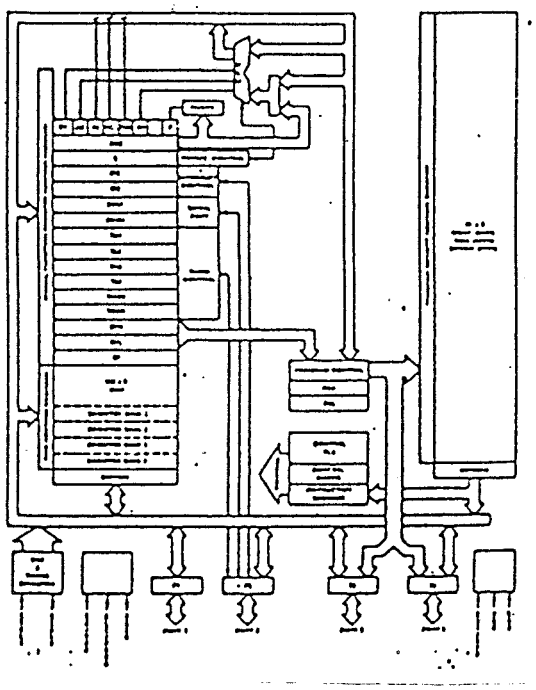
Fungsi dari tiap pin tersebut adalah :

1. PC (*program counter*) :

PC merupakan register 16 - bit yang digunakan untuk mengatur urutan pengambilan instruksi yang akan dijalankan.

2. Dekoder Instruksi :

Bagian ini menerjemahkan setiap instruksi program dan membangkitkan sinyal yang akan mengatur fungsi dari setiap bagian dari CPU.



Gambar 2.6 : Diagram Blok Mikrokontroler 8031[13]

3. RAM Data Internal :

RAM data internal yang dimiliki 8031 sebesar 128 byte yang terdiri dari :

- ***Register Bank*** **:** terdapat empat buah *register bank,* setiap *bank* terdiri dari 8 buah register (R0-R7).
- **128** ***Addressable Bits*** **:** bagian ini beralokasi pada alamat 20H sampai 2FH.

---

[13] Ibid, hal. 16

4. Register Fungsi Khusus (SFR) :

Register yang termasuk dalam register fungsi khusus ini adalah :

- **Register A :** register ini berfungsi sebagai registter akumulator.
- **Register B :** register ini digunakan bersama register A untuk instruksi perkalian dan pembagian.
- **Program Status Word (PSW) :**

register ini meliputi bit-bit : *carry* (CY), *auxilary carry* (AC), *zero flag* (FO), pemilih register bank (RS0 dan RS1), *over flow* (OV) dan *parity flag* (P). Flag CY, AC dan OV merupakan keluaran dari suatu proses aritmatika yang dilakukan di *accumulator. Flag* P merupakan *parity* dari isi register

accumulator. *Flag* C digunaakan pula sebagai *accumulator* untuk operasi bit. RS0 dan RS1 digunakan untuk memilih register *bank* yang aktif pada saat itu. Ada empat *bank* yang dapat dipilih untuk digunakan, semua bersifat *byte addressable.* Empat *bank* tersebut adalah :

RS1	RS0	REGISTER BANK
0	0	BANK 0
0	1	BANK 1
1	0	BANK 2
1	1	BANK 3

- **Penunjuk Stack (*stack poiter/SP*)**

SP adalah register 8 bit yang mrnunjukkan alamat data terakhir yang dimasukkan (*push*) ke dalam *stack,* juga sebagai alamat dari byte selanjutnya

yang akan dikeluarkan (*pop*). Nilai SP akan bertambah bila diberi instruksi *push* dan akan berkurang bila diberi instruksi *pop*. Lokasi SP dapat ditentukan secara perangkat lunak.

- **Port 0, 1, 2 dan 3**

Empat port 8 bit (32 bit) totalnya merupakan sarana berhubungan dengan dunia luar bagi mikrokontroler. Semua port dapat dialamati secara byte atau bit. Khusus port 0 dan port 2 dapat digunakan sebagai jalur data dan alamat untuk berhubungan dengan memori eksternal yang berkapasitas maksimal 64 KB. Port 3 dapat digunakan sebagai sinyal kontrol khusus, termasuk didalamnya terdapat pengirim dan penerima data secara serial. Satu-satunya port yang murni sebagai sarana masukkan dan keluaran adalah port 1.

- **Register Prioritas Interupsi (IPC)**

IPC berisi bit-bit kontrol untuk mengaktifkan interupsi sesuai dengan prioritas yang diinginkan.

- **Register Interupsi Enable (EIC)**

IEC digunakan untuk menyimpan bit-bit dan mengaktifkan kelima sumber interupsi dan menyimpan bit untuk menghidupkan/mematikan (*enable/disable*) semua interupsi.

- **Register Mode Waktu/Pencacah (TMOD)**

Register ini digunakan untuk memilih mode waktu atau pencacah yang akan digunakan.

- Register Kontrol Waktu/Pencacah (TCON)

TCON berisi kontrol terhadap penggunaan pewaktu/pencacah. Bit-bit mulai berhenti/mulai untuk pewaktu/pencacah, *flag-flag overflow* dan permintaan interupsi disimpan di register ini.

- Register-register Pewaktu/Pencacah 1 Tinggi dan Rendah (TH1 dan TL1) Pewaktu/Pencacah 0 Tinggi dan Rendah (TH0 dan TL0) :

Terdapat empat lokasi register untuk satu buah pewaktu/pencacah 16 bit dan dapat pula digunakan sebagai dua buah pencacah 8 bit. TH1 dan TH0 digunakan untuk 8 bit tinggi dari pewaktu /pencacah 1dan 0. TL1 dan TL0 digunakan 8 bit rendah dari pewaktu/pencacah 1 dan 0.

- Register Kontrol Serial (SCON)

SCON mempunyai fungsi utama untuk mengatur proses pengiriman dan penerimaan data serial. Pemilihan mode operasi serial dilakukan dalam register ini.

- Penyangga Data Serial (SBUF)

SBUF digunakan untuk menampung data yang akan dikirim atau data yang diterima melalui port serial.

Tabel 2.3: Register Fungsi Khusus (SFR) mikrokontroler 8031

Simbol	Nama	Alamat
ACC	Akumulator	0E0H
B	Register B	0F0H
PSW	Program Status Word	0D0H

Tabel berlanjut ke halaman berikut

Lanjutan Tabel : Register Fungsi Khusus (SFR)

Simbol	Nama	Alamat
SP	Penunjuk Stack	081H
DPTR	Penunjuk Data 2 Byte	082H
DPL	Byte Rendah	083H
DPH	Byte tinggi	084H
P0	Port 0	080H
P1	Port 1	090H
P2	Port 2	0A0H
P3	Port 3	0B0H
IP	Kontrol Prioritas Interupsi	0B8H
IE	Kontrol Pemungkin Interupsi	0A8H
TMOD	Kontrol Mode Pewaktu/Pencacah	089H
TCON	Kontrol Pewaktu/Pencacah	088H
TH0	Pewaktu/Pencacah 0 byte tinggi	08CH
TL0	Pewaktu/Pencacah 0 byte rendah	08AH
TH1	Pewaktu/Pencacah 1 byte tinggi	08DH
TL1	Pewaktu/Pencacah 1 byte rendah	08BH
SCON	Kontrol serial	088H
SBUF	Penyangga Data Serial	099H
PCON	Kontrol Power	087H

2.7.4 UNIT ARITMATIKA DAN LOGIKA (ALU)

ALU dapat melakukan operasi-operasi aritmatika dan fungsi-fungsi logika pada variabel-variabel 8 bit, seperti penambahan, pengurangan, perkalian dan pembagian, juga operasi-operasi logika AND, OR, serta fungsi-fungsi lainnya seperti *rotate, clear, complement* dan lain-lain. ALU juga dapat membuat keputusan kondisi suatu percabangan (*condition branching dicision*), dan

memberikan data *path* dan register-register sementara yang digunakan untuk transfer data dalam sistem. Instruksi-isntruksi lainnya dibuat dari fungsi-fungsi dasar ini.

Operasi-operasi dasar digabungkan dan dikombinasikan denga logika yang diperlukan untuk membuat instruksi-instruksi komples seperti *increment* register terpisah 16 bit. Sebagai contoh, untuk mengeksekusi satu bentuk instruksi *compare*, 8031 meng-*increment* program counter tiga kali, membaca tiga byte dari memory program, menghitung alamat register dengan operasi logika, dua kali membaca memori data internal, membuat perbandingan aritmatika dari dua buah variabel, menghitung 16 bit alamt tujuan dan memutuskan apakah melakukan suatu percabangan atau tidak yang semuanya ini hanya dilakukan dalam waktu dua mikrodetik.

ALU dapat memanipulasi data sama baiknya dengan 8 bit data. Bit-bit tunggal dapat di-*set, cleared, complemented,* dipindahkan, di-*test* dan digunakan dalam komputasi logika.

ALU dengan kemampuan yang tinggi ini menyebabkan 8031 dapat melekukan operasi kontrol secara *real time* dan algoritma data intensif. Operasi-operasi terpisah sebanyak 51 buah memindahkan dan memanipulasi tiga tipe data yaitu : *Boolean (1bit),* byte (8 bit) dan alamat (16 bit). Ada sebelas mode pengalamatan, yaitu tujuh untuk data dan empat untuk kontrol urutan program. Operai-operasi umumnya membolehkan beberapa mode pengalamatan.

2.7.5 RANGKAIAN OSILATOR

Pembangkit pewaktu telah lengkap terdapat dalam 8031, kecuali referensi frekwensi yang bisa berupa kristal atau sumber *clock eksternal.* Osilator yang tersedia adalah rangkaian paralel anti-resonansi. Frekwensi tersebut dibagi 12 oleh pewaktu internal yang memberikan 8031 siklus instruksi minimum 1 uSc dengan kristal 12 MHz. Pin XTAL2 adalah output dari amplifier berpenguat tinggi. Sedangkan XTAL1 adalah inputnya. Hubungan kristal antara XTAL1 dan XTAL2 memberikan umpan balik da *phase shift* yang diperlukan untuk berosolasi. Jika XTAL1 dikemudikan oleh sumber frekwensi ekternal, XTAL2 tidak dihubungkan. Frekwensi 1,2 MHz sampai 12 MHz juga diperbolehkan bila digunakan sumber frekwensi luar sebaga *clock* pada XTAL1

2.7.6 PEWAKTU CPU

Satu *machine cycle* terdiri dari 6 keadaan (12 periode osilator) dan setiap keadaan dibagi dalam dua fase yang berhubungan denga dua fase sinyal *clock.* Secara normal operasi-operasi aritmatika dan logika dilakukan pada fase pertama dan transfer register ke register internal dilakukan pada fase dua.

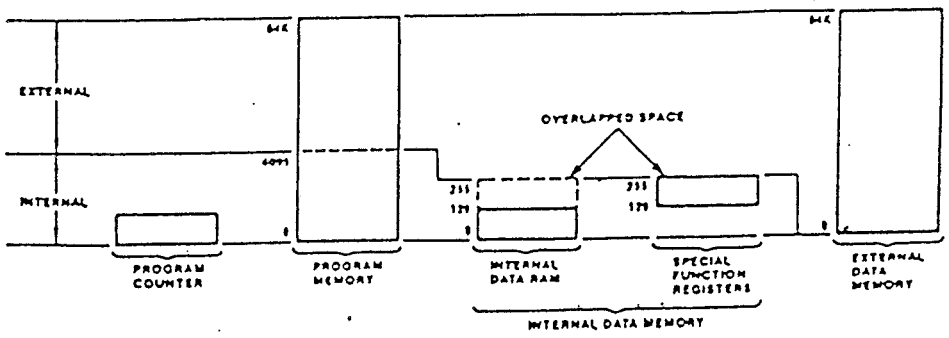
Karena sinyal *clock* ini tidak dapat diamati dari luar, maka sinyal XTAL2 dan ALE dapat dipakai sebegai eferensi ekternal. Satu *machine cycle* terdiri dari 12 periode osilator, diberi nomor S1P1 (*state 1 phase*) sampai S6P6. Setiap state memiliki durasi selama dua periode osilator. ALE aktif dua kali setiap *machine cycle*, sekali selama S1P2 dan S2P1, dan sekali lagi selama S2P2 dan S5P1.

Eksekusi dari instruksi *one-cycle* mulai pada S1P2. Jika mengeksekusi instruksi dua byte, byte yang kedua dibaca selama S4 dari *machine cycle* yang sama. Jika merupakan instruksi satu byte, akan dibaca pada S4, tetapi pembacaan byte (*opcode* berikutnya) diabaikan, dan program counter tidak dinaikkan. Pada umumnya eksekusi selesai pada akhir dari S6P6. Kebanyakan instruksi-instruksi dieksekusi dalam satu *cycle*. Hanya instruksi perkalian memerlukan 4 *cycle*.

2.7.7 MEMORI

2.7.7.1 Organisasi Memori

Dalam 8031 memori diorganisasikan atas tiga ruang alamat dan program *counter*. Gambar 2.7 memperlihatkan organisasi memori dari 8031.



Gambar 2.7 : Organisasi Memori 8031[14]

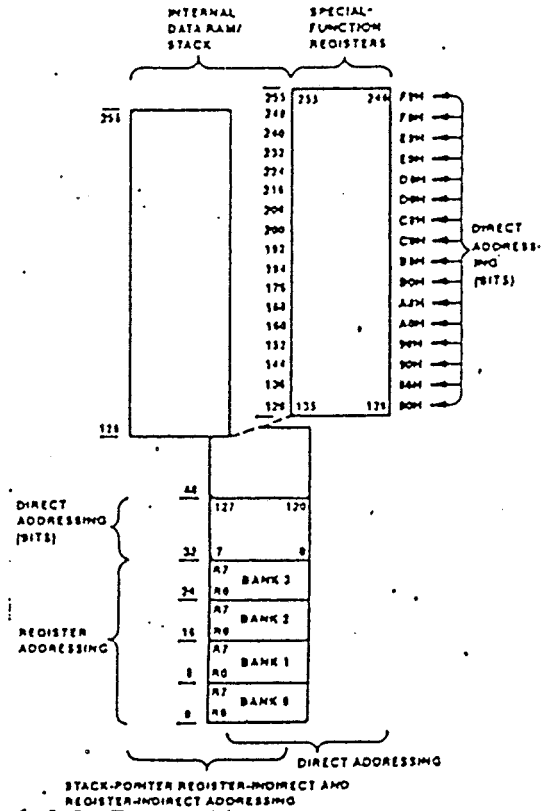
- 16 bit counter
- 64 kilo byte ruang alamat memori program
- 64 kilo byte ruang alamat memori data ekternal

---

[14] Ibid, hal.17

Register *program counter* 16 bit memperlengkapi 8031 denga kemampuan pengalamatan 64 kilo byte. *Program counter* memungkinkan pemakai untuk mengeksekusi perintah dan percabangan pada setiap lokasi dalam ruang memori program. Tidak ada instruksi yang mengizinkan eksekusi program untuk memindahkan ruang memori program ke setiap ruang dari memori data.

Lokasi-lokasi tertentu dalam memori program disediakan untuk program-program khusus. Lokasi 0000 sampai 0002 disediakan untuk program inisialisasi. Setelah *reset* dilakukan, CPU selalu mulai dengan mengeksekusi program pada lokasi 0000. Lokasi 0003 sampai 0042 disediakan untuk pelayanan lima buah permintaan interupsi. Ke-64 kilo byte alamat ruang memory data eksternal secara otomatis diakses ketika instruksi MOVX dilaksanakan.



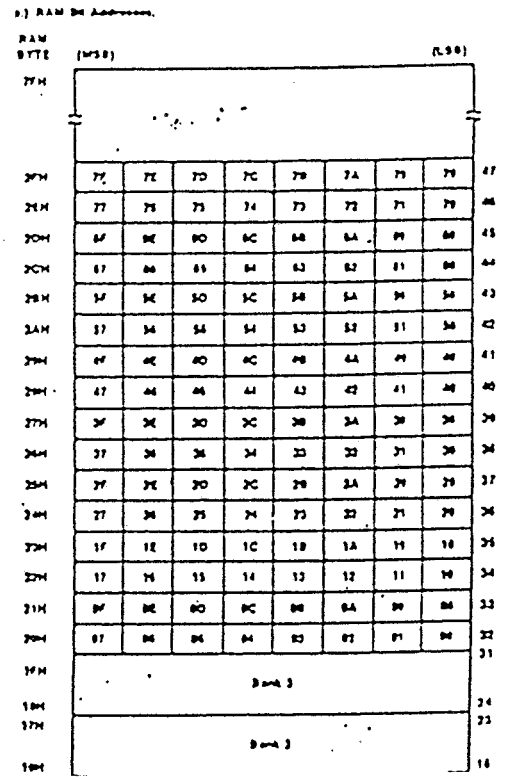
Gmb 2.8 : Ruang Alamat Memori Data Internal[15]

---

15 Ibid, hal. 18

Secara fungsional, memori data internal adalah ruang alamat yang paling fleksibel, seperti memori data internal. Ruang memori data internal dan 128 byte ruang alamat register-register fungsi khusus (SFR). Dalam ruang alamat ini terdapat 256 bit-bit terpisah yang diamati. Gambar 2.8 menunjukkan lokasi-lokasi dari ruang-ruang alamat memori data internal.

Ruang alamat RAM data internal adalah 0 sampai 255. Empat *banks* dari delapan register menempati lokasi-lokasi 0 sampai 31. *Stack* dapat menempati setiap lokasi dalam ruan alamat RAM data internal. Sebagai tambahan, 128 lokasi dari RAM data internal dapat diakses melalui pengalamatan langsung (*direct addressing*). Bit-bit ini terletak di dalam RAM data internal pada lokasi-lokasi 32 byte sampai 47, seperti terlihat pada gambar 2 9

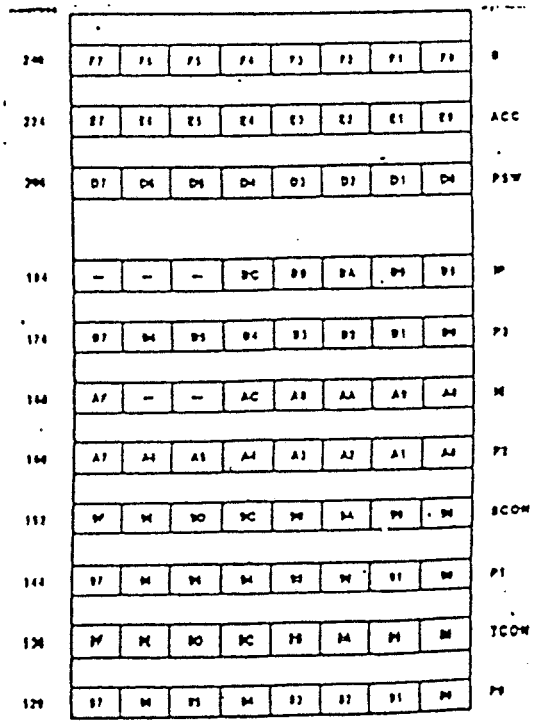


Gmb 2.9 : Alamat-alamat Bit RAM Internal[16]

16 Ibid, hal. 18

Banyak *stack* hanya dibatasi oleh kemampuan RAM data internal. *Stack* digunakan untuk menyimpan *program counter* selama proses pemanggilan subroutine, dan bisa digunakan untuk menyimpan parameter-parameter. Setiap byte dari RAM data internal atau SFR dapat diakses melalui pengalamatan, langsung dapat disimpan atau dikeluarkan (*pushed/poped*)

Ruang alamat register-register fungsi khusus adalah 128 sampai 255. Semua register selain *program counter* dan empat *banks* dari delapan register-register fungsi khusus memungkinkan mereka dapat diakses langsung seperti RAM internal. Mereka dapat diakses oleh hampir semua variabel.



Gambar 2.10 : Alamat-alamat Register Fungsi Khusus[17]

[17] Ibid, hal. 19

Sebagai tambahan 128 bit lokasi-lokasi dalam ruang alamat register-register fungsi khusus dapat diakses melalui pengalamatan langsung.Bit-bit ini berada dalam ruang alamat register-register fungsi khusus dan dapat diakses menggunakan pengalamatan langsung. Dua puluh register-register fungsi khusus ada pada tabel 2.4, dan pemetaan register-register fungsi khusus ada pada gambar 2.10.

Tabel 2.4

Register-register Fungsi Khusus

SPECIAL FUNCTION REGISTER		ASM-51	LOCATION
Arithmetic Register	Accumulator	ACC	224 (EOH)
	B register	B	240 (FOH)
	Program Status- Word	PSW	208 (DOH)
Pointers	Stack Pointer	SP	129 (81H)
	Data Pointer (High)	DPH	131 (83H)
	Data pointer (Low)	DPL	130 (82H)
Pararel I/O Port	Port 3	P3	176 (BOH)
	Port 2	P2	160 (AOH)
	Port 1	P1	144 (90H)
	Port 0	P0	128 (80H)
Interrupt System	Interrupt Priority Control	IPC	184 (B8H)
	Interrupt Enable Control	IEC	183 (A8H)
Timers	Timer Mode	TMOD	137 (89H)
	Timer Control	TCON	136 (88H)
	Timer 1 (high)	TH1	141 (8DH)
	Timer 1 (low)	TL1	139 (8BH)
	Timer 0 (high)	TH0	140 (8CH)
	Timer 0 (low)	TL0	138 (8AH)
Serial I/O Channel	Serial Control	SCON	152 (98H)
	Serial Data Buffer	SBUF	153 (99H)

2.7.7.2 MODE-MODE PENGALAMATAN

Karena arsitektur 8031 membedakan antara memori data dengan program memori, maka masing-masing mempunyai mode pengalamatan yang berbeda-beda. Ada lima mode pengalamatan *source operand,* yaitu :

- *register addressing*
- *direct addressing*
- *register indirect addressing*
- *immediate addressing*
- *base register plus index register indirect addressing*

Tiga mode yang pertama juga dapat berfungsi untuk alamat suatu *destination operand.* Pada umumnya instruksi-instruksi mempunyai *destination source* yang menspesifikasikan tipe data, mode pengalatan dan *operand-operand* yang diperlukan. Selain untuk instruksi mov, *distenition operand* juga berfungsi sebagai *source operand* yang berlokasi di memori data internal. Pemilihan antara memori program dan memori data ekternal ditentukan oleh *menemonic,* kecuali untuk *immediate operand.*

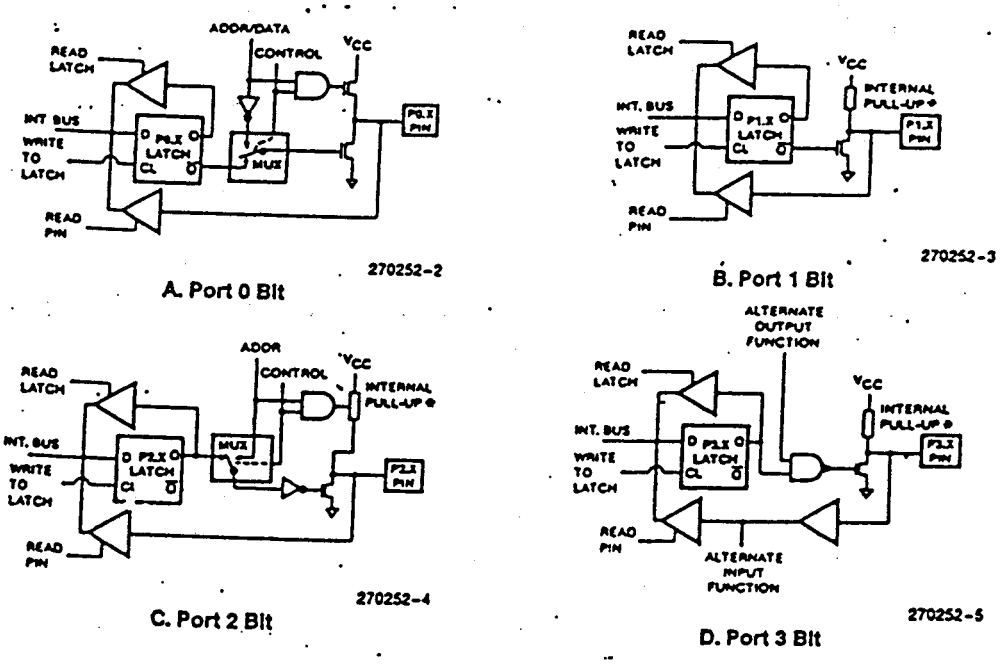
2.7.8 OPERASI INPUT/OUTPUT

Setiap pin dari 32 jalur I/O yang terbagi menjadi empat port delapan bit, masing-masingnya dapat diprogram secara terpisah sebagai input atau output dan masing-masing dapat dikontrol melalui perangkat lunak. Untuk memfungsikan suatu port sebagai input, dapat dilakukan dengan menuliskan logika '1' pada

masing-masing pin. Setiap kali suatu instruksi menggunakan Port sebagai tujuan, harus ditulis '1' pada bit-bit yang berhubungan dengan pin-pin input. Suatu input ke sebuah port tidak memerlukan sinkronisasi dengan osilator. Setiap pin port di-*sample* pada akhir dari sinyal ALE selama instruksi pembacaan.

Port 0 mempunyai *output bidirectional open-drain*. Bila digunakan sebagai bus, port 0 mempunyai pengendali tiga kondisi standar (*standar three status driver*), seperti pada gambar 2.11.

Port 1, 2 dan 3 mempunyai driver output *quasi bidirectional* yang dihubungakan dengan tahanan *pull-up* sebesar 10K ohm sampai 20K ohm seperti terlihat pada gambar 2.11.



Gambar 2.11 : Struktur Port 0, 1, 2 dan 3

2.7.9 MENGAKSES MEMORI EKTERNAL

Mikrokontroler 8031 dapat mengakses 64 kilo byte memori data ekternal dan 64 kilo byte memori program ekternal. Pengaksesan memori ekternal dapat dilakukan bila pin AE rendah. Sinyal ALE, PSEN, RD dan WR digunakan sebagai sinyal kontrol memori. Sinyal ALE digunakan untuk menahan alamat ke memori ekternal. Sinyal PSEN (*program store enable*) digunakan untuk mengakses memori program ekternal. Sedangkan sinyal RD dan WR digunakan untuk mengakses memori data ekternal.

Pada saat mengakses memori ekternal, mikrokontroler 8031 mengeluarkan byte alamat tinggi melalui port 2 dan byte alamat rendah (juga data) melalui port 0. Setiap siklus bus memori program terdiri dari enam periode osilator. Setiap periode dinamakan T1 sampai T6. Alamat dikeluarkan dari mikrokontroler selama T3. Tranfer data terjadi selama pada bus selama T5, T6. Dan siklus bus berikutnya adalah T1.

Siklus pembacaan mulai pada T2 dengan adanya sinyal ALE. Pada akhir dari sinyal ALE digunakan untuk melewatkan informasi alamat yang ada di bus saat itu. Pada T5, alamat dihilangkan dari bus port 0 dan pengendalian bus berada pada keadaan impedansi tingggi. Kontrol pembacaan memori program juga muncul pada T5. PSEN memnyebabkan kompnen yang dialamati meng-*enable* pengendali bus-nya, sehingga tidak lama kemudian data instruksi yang diinginkan telah berada pada bus. Ketika 8031 mengembalikan sinyal PSEN pada keadaan tonggi, pengendali bus dari komponen yang dialamati akan mengambang.

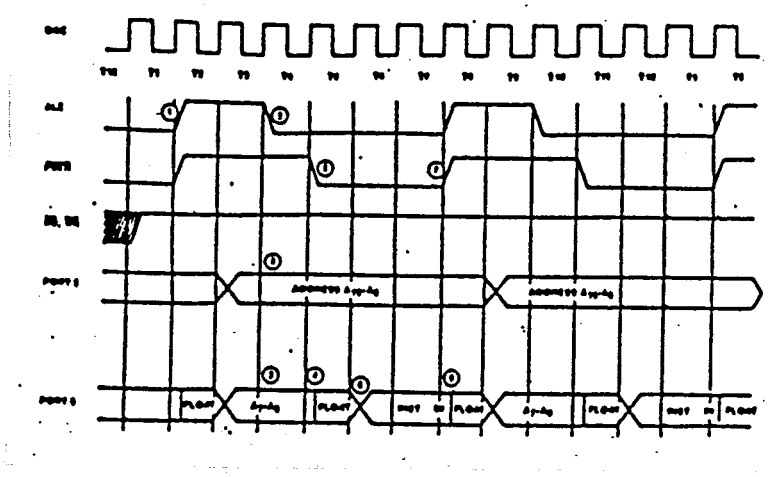
Setiap siklus bus memori data eksternal terdiri dari dua belas periode osilator, yang diberi nama T1 sampai T12. Alamat dikeluarkan dari prosesor selama T3. Tranfer data terjadi pada bus selama T7 sampai T12. T5 dan T6 adalah periode dimana arah dari bus diubah untuk operasi pembacaan. Siklus pembacaan dimulai pada T2 dengan adanya sinyal ALE. Pada akhir sinyal ALE digunakan untuk melewatkan informasi alamat yang ada pada bus. Pada T5, alamat dihilangkan dari bus dan bus port 0 berada pada keadaan impedansi tinggi. Sinyal kontrol pembacaan memori RD, keluar selama T7. Sinyal RD menyebabkan komponen yang dialamati meng-*enable* pengendali bus-nya. Tidak lama kemudian data yang diinginkan akan berada pada bus. Pada saat 8031 mengembalikan sinyal RD pada keadaan tinggi, pengendali bus pada komponenyang dialamati akan mengambang.

Siklus pembacaan juga dimulai dengan sinyal ALE ddan pengiriman alamat. Pada T6, prosesor mengirim data untuk ditulis ke dalam lokasi data memori yang dialamati. Data ini tetap ada pada bus sampai akhir siklus bus berikutnya T2. Sinyal tulis WR menjadi rendah pada T6 dan tetap aktif sampai T12.

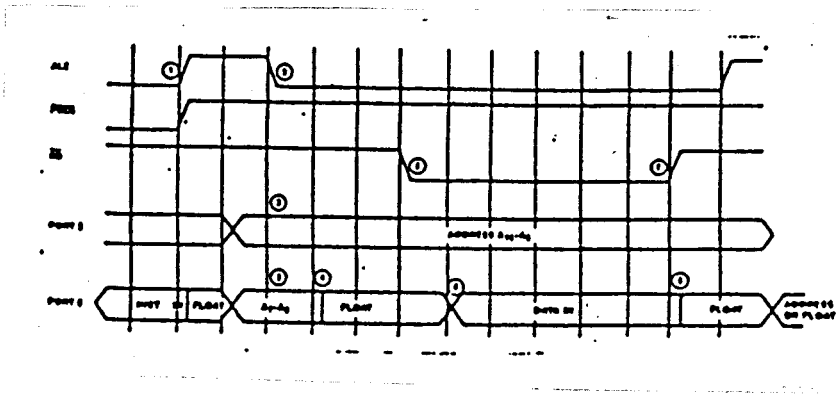
Diagram waktu dari proses pembacaan memori program, pembacaan memori data dan penulisan memori data diperlukan secara berturut-turut pada gambar 2.12, gambar 2.13 dan gambar 2.14.

2.7.10 SISTIM INTERRUPT

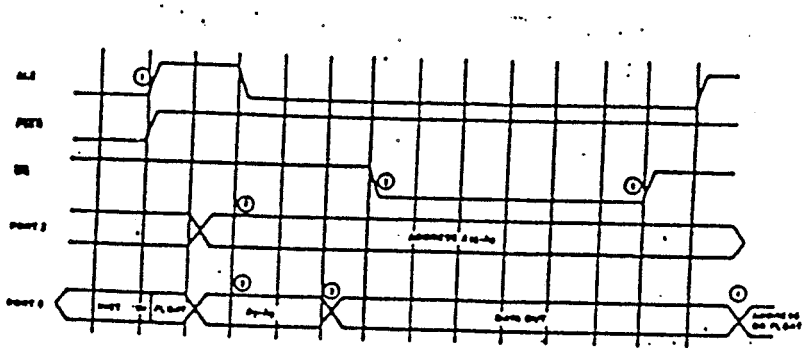
Interrupt berfungsi untuk mentransfer kontrol program ke suatu lokasi program baru. Pada 8031 terdapat lima sumber perangkat keras yang dapat membangkitkan suatu permintaan interrupt.



Gambar 2.12 : Siklus Waktu Pembacaan Memori Program



Gambar : 2.13 Siklus Waktu Pembacaan Memori Data



Gambar 2.14 : Siklus Waktu Penulisan Memori Data

Setiap interrupt dapat diaktifkan atau tidak secara terpisah dengan mengubah bit '1' atau '0' pada register *interrupt enable* (IE). Semua sumber interrupt juaga dapat diaktifkan secara keseluruhan yaitu oleh bit EA dari register IE.

Sumber-sumber interrupt dan alamat awal dari *vector address* pelayanan interupt terlihat pada tabel 2.5. Susunan bit-bit dari register IE digambarkan pada gambar 2.15. Posisi bit-bit pada register, nama dan artinya ditunjukkan pada tabel 2.6. Sedangkan untuk register IP pada gambar 2.16 dan tabel 2.7. Disamping level prioritas diatas, terdapat struktur prioritas kedua yang ditentukan oleh urutan *polling* yaitu sebagai berikut (dari yang tinggi ke yang rendah) : IEO, TFO, IEI dan R1+T1, dengan catatan struktur prioritas ini hanya digunakan untuk menyelesaikan permintaan yang serentak dari level prioritas yang sama.

(MSB) (LSB)

EA	--	--	ES	ET1	EX1	ETO	EXO

Gambar 2.15 Register Interrupt Enable

(MSB) (LSB)

--	--	--	PS	PT1	PX1	PT0	PX0

Gambar 2.16 Register Interrupt Priority

Tabel 2.5 : Alamat Awal Program Pelayanan Interrupt

NO	SUMBER INTERRUPT	ALAMAT AWAL
1	Permintaan Eksternal 0	3 (0003H)
2	Timer/Counter Internal 0	11 (000BH)
3	Permintaan Eksternal 1	19 (0013H)
4	Counter/Timer Internal 1	27 (001BH)
5	Port Serial Internal	35 (0023H)

Tabel 2.6 : Nama dan Arti IE - Interup Enable Register

SIMBOL	POSISI	NAMA DAN ARTI
EA	IE.7	*Enable Al*. Bila EA=0, semua interrupt dimatikan, tidak tergantung keadaan IE.0 - Ie.4.
ES	IE.4	*Enable Serial Port*. Bila ES=0, interrupt priority dimatikan.
ET1	IE.3	*Enable Timer* 1. Bila Et=0, interrupt pewaktu 1 dimatikan.
EX1	IE.2	*Enable External Interrupt* 1. Bila EX1=0 *External Interrupt* 1 dimatikan.
ET0	IE.1	*Enable Timer* 1. Bila ET=0, interrupt pewaktu 1 dimatikan.
EX0	IE.0	*Enable External Interrupt* 0. Bila EX=0, *External interrupt* 0 dimatikan.

Register TCON digunakan untuk menentukan *trigger* dari interupt eksternal, sisi jatuh (*edge triggered*) atau taraf rendah, menjalankan atau menghentikan pewaktu/pencacah.

Tabel 2.7 : Nama dan arti Interupt Priority Register

SIMBOL	POSISI	NAMA DAN ARTI
PS	IP.4	*Serial port priority level.* Bila PS=1 prioritasnya menjadi lebih tinggi.
PT1	IP.3	*Timer 1 priority level.* Bila PT1=1 tingkat prioritasnya lebih tinggi.
PX1	IP.2	*Exsternal interupt 1 priority level.* Bila PX=1 prioritasnya lebih tinggi.
PT0	IP.1	*Timer 0 priority level.* Bila PT0=1 prioritasnya lebih tinggi.
PX0	IP.0	*External interupt 0 priority level.* Bila PX0=1 tingkat prioritasnya menjadi lebih tinggi.

TImer overflow flag pada register ini akan di-set oleh perangkat keras jika pewaktu/pencacah *overflow* dan *interupt edge flag* di-set bila terdeteksi perubahan dari logika '1' ke logika '0' dari sinyal interupt luar. Susunan bit-bit dari register TCON digambarkan pada gambar 2.17. Posisi bit-bit pada register, nama dan artinya ditunjukkan pada tabel 2.8.

2.7.11 PEWAKTU/PENCACAH (*TIMER/COUNTER*)

Mikrokontroler 8031 mempunyai dua buah register 16 bit yang dapat digunakan sebagai pewaktu/pencacah. Register tersebut adalah TH0/TL0 (pewaktu / pencacah 0 byte tinggi / rendah). Setiap pewaktu/pencacah dikontrol oleh bit dalam register TMOD untuk memilih fungsi sebagai pewaktu/pencacah.

(MSB) (LSB)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Gambar 2.17. Register *timer/counter control/status* TCON

Terdapat empat mode pewaktu /pencacah yang dapat dipilih, yaitu :

- Mode 0 :

Dalam mode ini pewaktu/pencacah dikonfigurasikan sebagai register 13 bit dengan *prescaler* pembagi 32. Pewaktu/pencacah 1 bekerja apabila TR1=1 dan bila GATE=0 atau INT1=1, begitu pula untuk pewaktu/pencacah 0.

- Mode 1 :

Mode 1 sama dengan mode 0, kecuali register timer difungsikan sebagai 16 bit.

- Mode 2 :

Menghasilkan pewaktu/pencacah 8 bit dengan *automatic reload.* Register TH1 dan TH0 berisi bilangan yang akan *direload* ke register tl1 dan TL0 setiap kali *overflow.*

Tabel 2.8. Nama dan arti register TCON

SIMBOL	POSISI	NAMA DAN ARTI
TF1	TCON.7	*Timer 0 overflow flag.* Disini '1' oleh perangkat keras bila pewaktu/pencacah overflow. Disini '0' bila interupt diproses.
TR1	TCON.6	*Timer I run control bit.* Diset oleh perangkat lunak untuk menjalankan/menghentikan pewaktu / penacacah.
TF0	TCON.5	*Timer o overflow flag.* Diset oleh perangkata keras bila pewaktu/pencacah *overflowI.* Di-reset bila interup diproses.

Tabel berlanjut ke halaman berikut.

Lanjutan Tabel : Nama dan arti register TCON

SIMBOL	POSISI	NAMA DAN ARTI
TR0	TCON.4	*Interupt 0 run control bit.* Di-set/reset oleh perangkat lunak untuk menjalankan / menghentikan pewaktu/pencacah.
IE1	TCON.3	*Interupt 1 byte flag.* Di-set oleh perangkat keras bila sisi interupt luar diproses.
IT1	TCON.2	*Interupt 1 type control bit.* Di-set/reset oleh perangkat lunak untuk menentukan *falling edge/low level trigger* dari luar.
IE0	TCON.1	*Interupt 0 edge interupt.* Di-set oleh hardware bila sisi interupt luar terdeteksi. Di-set bila interupt diproses.
IT0	TCON.0	*Interupt o type control bit.* Di-set/reset oleh software untuk menentukan *falling edge/low level trigger* dari interupt luar.

- Mode 3 :

Pewaktu 0 dibuat menjadi dua buah pewaktu/pencacah 8 bit yang terpisah. Pewaktu 1 tidak bekerja. Konfigurasi *hardware* dari berbagai mode diperlihatkan dalam gambar 2.18. Susunan bit-bit dari register TMOD digambarkan pada gambar 2.19. Fungsi dari masing-masing bit adalah sebagai berikut :

1. Gate :

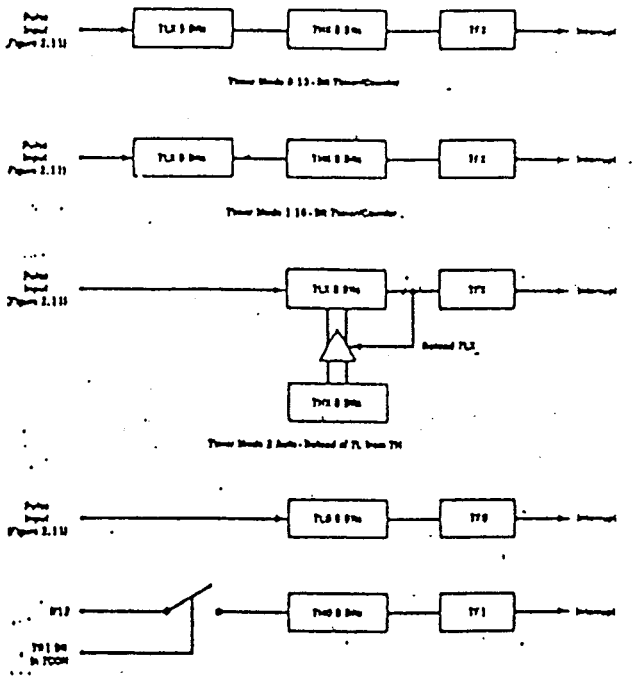
Kontrol gerbang. Bila GATE=1, pewaktu 0 atau pewaktu 1 akan diaktifkan hanya bila kaki INT0/INT1 mendapat masukkan tinggi dan bit kontrol TR0/TR1 = 1. Apabila GATE=0 pewaktu 0 atau pewaktu 1 diaktifkan apabila TR0/TR1 = 1

2. C/T :

Pemilihan operasi pewaktu/pencacah. Bila C/T = 0, untuk operasi pewaktu (masukkan dari sistem clock di dalam), dan bila C/T = 1 untuk operasi pencacah (masukkan dari kaki T0/T1).

3. M0/M1 :

Pemilihan mode operasi. M1=0 dan M1=0 untuk mode 0, M1=0 dan M0=1 untuk mode 1, M1=1 dan M0=0 untuk mode 2 dan M1=1 dan M0=1 untuk mode 3.



Gambar : 2.18 Model timer/counter

(MSB) (LSB)

GATE	C/T	M1	M0	GATE	C/T	M1	M0

Gmb 2.19 : Register *Timer/counter* mode (TMOD)

2.7.12. PORT SERIAL

Port serial digunakan untuk komunikasi data serial, baik yang menggunakan hubungan *half duplex* maupun *full duplex,* atau untuk menambah port I/O dengan menggunakan register geser (*shift register*). Port serial dapat dioperasikan dalam empat mode, yaitu :

- Mode 0 (*synchronous*) :

Data serial 8 bit dikirim dan diterima melaui RxD denga bit terendah (LSB) yang pertama, dan TxD mengeluarkan clock penggeser (*shift clock*). Boud rate adalah 1/12 frekwensi osilator.

- Mode 1 (*asynchronous*) :

Sepuluh bit ditransmisikan melaui TxD atau diterima melaui RxD dengan ukuran : 1 *start bit,* 8 bit data (LSB yang pertama) dan 1 bit. *Boud rate-nya* variabel.

- Mode 2 :

Sebelas bit ditransmisikan melalui TxD atau diterima melaui RxD dengan urutan : 1 *start bit,* 8 bit data (LSB) yang pertama, 1 bit data yang dapat diprogram dan 1 *stop bit.* Pada pengiriman data, bit data ke 9 (TB8) dapat dipilh 1 atau 0. TB8 dapat digunakan sebagai *parity bit.* Pada penerima data, bit data ke 9 akan mengisi RB8 pada register SCON dan *stop bit* diabaikan. *Boud rate* dapat diprogram untuk 1/32 atau 1/64 frekwensi osilator.

- Mode 3 :

Mode 3 sama dengan mode 2, kecuali pada mode 3 *boud rate-nya* dapat diubah-ubah (*variabel*).

2.7.12.1 BAUD RATE

Baud rate untuk mode 0 adalah 1/12 frekwensi osilator. *Baud rate* untuk mode 2 bergantung pada nilai bit SMOD didalam register fungsi khusus PCON. Jika SMOD=0 maka *baud rate-nya* adalah 1/64 dari frekwensi osilator. Jika SMOD=1, *baud rate-nya* adalah 1/32 dari frekwensi osilator.

Baud rate untuk mode 1 dan 3 ditentukan oleh *overflow rate timer* 1 dan *baud rate-nya* ditentukan oleh rumus sebagai berikut :

$$\text{Baud rate} = \frac{2^{\text{SMOD}}}{32} \times \frac{\text{Frekwensi osilator}}{12 \times [256-(\text{TH1})]}$$

TH1 adalah nila *reload* yang diisikan dengan perangkat lunak pada register TH1.

2.7.12.2 REGISTER SERIAL PORT CONTROL (SCON)

Register SCON digunakan untuk mendefinisikan mode operasi dan kontrol fungsi-fungsi tertentu dari port serial. Register ini juga menerima bit data ke 9 (RB8) dan mengirim serta menerima *interupt flag* (T1 dan R1). Register SCON dibamgarkan pada gambar 2.17.

SM0 dan SM1 digunakan untuk memilih mode operasi dari port serial seperti yang ditunjukkan pada tabel 2.7. Sedangkan fungsi bit-bit lainnya adalah :

- SM2 :

Pada mode 2 atau 3, bila SM2=1 maka RI tidak akan diaktifkan bila bit data ke 9 (RB8) yang diterima adalah 0. Pada mode 1, jika SM2=1 maka RI tidak akan

diaktifkan bila *stop bit* yang sama sempurna tidak diterima. Pada mode 0 SM2 tidak digunakan.

- REN :

Bit ini di-set secara sotf ware untuk menjalankan atau menhentikan penerimaan data.

(MSB) (LSB)

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Gambar 2.20 : Register SCON

- TB8 :

Merupakan data bit yang ke 9 yang akan dikirim pada mode 2 dan 3. Di-set atau clear secara software.

- RB8 :

Pada mode 2 dan 3, merupakan bit data ke 9 yang diterima. Pada mode 1, jika SM2=0 RB8 adalah *stop bit* yang diterima. Pada mode 0 RB8 tidak dugunakan.

- TI :

Sebagai *transmit interupt flag* yang di-set secara *hardware* pada akhir dari waktu bit yang ke 8 pada mode 0, atau pada awal dari *stop bit* dalam mode yang lainnya. Dalam pengiriman secara serial, TI harus *di-clear* secara *sotfware.*

- RI :

Merupaka *receive interupt flag* yang di-set secara hardware pada akhir dari waktu bit ke 8 pada mode 0, atau pada setengah dari pemasukkan *stop bit* pada mode yang lainnya. Pada penerimaan serial, RI harus *di-clear* secara *software.*

2.7.12.3 SERIAL DATA BUFFER REGISTER (SBUF)

SBUF merupakan buffer untuk pengiriman data / penerima data secara serial. Register SBUF untuk pengiriman adalah berupa register geser 9 bit. Proses menulis data ke SBUF, juga berarti menulis bit ke 9 dari register geser dengan 1 atau TB8, bergantung dari mode yang dipilih.

Tabel 2.9. Pemilihan mode operasi bit serial

SM0	SM1	SM2	FUNGSI	BAUD RATE
0	0	0	register geser	f_{osc} / 12
0	0	1	8 bit UART	variabel
1	0	2	9 bit UART	f_{osc} / 64 atau f_{osc} / 32
1	2	3	9 bit UART	variabel

Register-register penerima merupakan register geser dengan panjang 8 bit pada mode 0, atau 9 bit pada mode yang lain, pada register SBUF dan sebuah *read only register* yang diisi oleh perangkat keras dengan byte data. Pada saat yang bersamaan dengan aktifnya RI. Pada mode UART, bit ke 9 diisikan ke RB8 pada register SCON pada saat yang bersamaan dengan pengisian byte data ke dalam SBUF.

2.7.12.4 PROGRAM STATUS WORD REGISTER (PSW)

Register status program (PSW) digunakan untuk menyimpan keadaan (status) dari mikrokontroler dan kontrol operasi untuk mikrokontroler. Gambar register PSW ditunjukkan pada gambar 2.21. Berikut adalah arti dari masing-masing bit dari PSW :

- CY (*carry flag*) :

Di-set atau clear secara *hardware* atau *software* sepanjang instruksi aritmatika atau logika tertentu.

- AC (*auxilary flag*) :

Di-set atau clear sepanjang instruksi penambahan atau pengurangan untuk menunjukkan adanya *carry.*

- F0 (*flag 0*) :

Di-set/clear atau dites secara *software* sebagai status *flag* yang didefinisikan oleh pemakai.

- RS0 (*register bank select control bit 0*) :

Di-set/clear secara *software* untuk menentukan *register bank* yang bekerja.

- RS1 (*register bank select control bit 1*) :

Di-set/clear secara *software* untuk menentukan *register bank* yang bekerja.

- OV (overflow flag) :

Di-*set* atau *clear* sepanjang instruksi aritmatika secara *hardware* untuk menunjukkan kodisi *overflow.*

- P (parity flag) :

Di-*set/clear* secara *hardware* setiap daur instruksi untuk menunjukkan kondisi *odd / even* dari bit 1 dari akumulator.

(MSB) (LSB)

CY	AC	PO	RS1	RS0	OV	-	P

Gambar 2.21 Register PSW

2.8 CodeBase

CodeBase berdasar pada pemrograman C standar. Tiap source program atau modul CodeBase, mengandung Function-Function CodeBase dan CodeBase struktur.

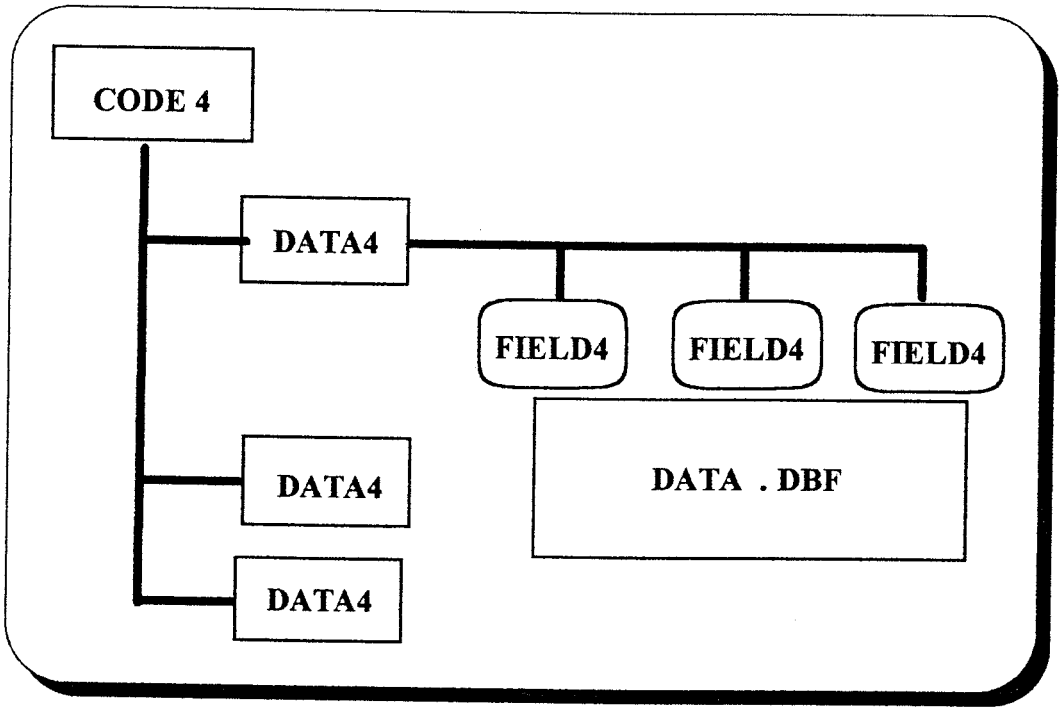
2.8.1 Unsur CodeBase (Function, Structure, Constant)

Untuk seluruh manipulasi data, Function-Function CodeBase selalu dalam nama Lower Case diawali dengan satu sampai enam huruf yang diawali dengan angka 4. Tulisan (huruf) awal mengindikasikan modul dari Function, yang sesuai dengan fungsi rutin yang bersangkutan. Contohnya, Function yang memanipulasi Data File dimulai dengan kode **d4**, sedangkan seluruh Function yang memanipulasi penanggalan dimulai dengan perintah **date4**.

Strukture (Structure) CodeBase digunakan untuk menyimpan informasi dan referensi objek-objek, seperti Data File dan Field. Struktur CodeBase diikuti

dengan konvensi penamaan yang sama seperti Function-Function CodeBase, kecuali yang dalam bentuk Upper Case.

Bentuk lain selain struktur dan Function dalam CodeBase adalah CodeBase Constant. Kebanyakan Constant di CodeBase dalam Integer yang biasanya adalah harga Return dan kode Error. Constant ini mempunyai nama-nama yang selalu diawali dengan **r4**, atau **e4**.



Gambar 2.22 Struktur CodeBase[18]

Sebagai pusat dari seluruh struktur ini adalah struktur **CODE4**. Struktur ini berisi setting-setting dan informasi yang dipakai dalam Function CodeBase.

---

[18] Sequiter, CodeBase 5.0 User's Guide, Sequiter Software Inc., hal. 23

Dalam CodeBase, hanya dibutuhkan satu buah struktur **CODE4** untuk seluruh aplikasinya. Bila dipakai lebih dari satu struktur **CODE4**, akan terjadi pemborosan pemakaian memori.

Struktur **DATA4** digunakan untuk referensi sebuah Data File khusus. Ketika sebuah Data File dibuka, sebuah struktur **DATA4** dialokasikan dan sebuah catatan refernsi tentang itu ditulis di struktur **CODE4**. Ketika memanggil sebuah Function yang bekerja pada sebuah Data File, harus dispesifikasikan Data File mana yang akan dihubungkan dengan Function sebuah Pointer di Struktur **DATA4**.

Tiap **DATA** sruktur mempunyai daerah kerja memori yang sama, yang disebut 'Record Buffer', yang mana digunakan untuk menyimpan sebuah Record dari Data File. Record yang disimpan dalam Record Buffer disebut 'Current Record'.

Tiap kali terjadi perubahan di Record Buffer, dengan dipakainya Function Field, akan otomatis ditulis ke Data File sebelum Record yang lain dibaca.

Struktur **FIELD4** digunakan sebagai referensi sebuah Field yang khusus dalam Record Buffer. Ketika Data file dibuka, sebuah **FILED4** otomatis terbentuk untuk tiap Field-nya. Ketika sebuah Function yang mempergunakan sebuah Field tertentu dari Record Buffer dipanggil, harus ditentukan Field mana yang memakai pointer **FIELD4**.

Current Record adalah Record yang berada dalam Record Buffer. Dan Record Buffer adalah sebuah buffer yang mengandung satu record dari Data File.

2.8.2 Inisialisasi CodeBase

Karena ketika program dimulai data yang dikandung struktur **CODE4** acak, struktur **CODE4** harus di-inisialisasi sebelum Function CodeBase apapun digunakan. Hal ini dilakukan dengan mem-passing pointer dari struktur tersebut dengan Function **d4init**.

Sehingga untuk inisialisasi dipanggil demikian :

```
d4init(&Code_Base);
```

Saat struktur **CODE4** di-inisialisasi, seluruh Flag dam variabel Member akan di-set ke harga default. Function **d4init** normalnya hanya diperkenankan dipanggil satu kali dari seluruh aplikasi.

2.8.3 Open Dan Close Data File

Hanya setelah **CODE4** di-inisialisasi, data file bisa dibuka dengan Function **d4open()**. Function **d4open** ber-input string yang mengandung nama file dari Data File. Bila nama yang di-spesifikasikan tidak mengandung extensien sama sekali, default extensien ".DBF" akan dipakai. Bila tak ada spesifikasi path-nya, akan dicari di direktori Current. Pemanggilan Function-nya :

```
Data_File = d4open(*Code_Base, argv[1]);
e4exit_test(&Code_Base);
```

Bila file ditemukan, file segera dibuka dan alamat pointer-nya dikembalikan ke struktur **DATA4**. Alamat ini digunakan sebagai sebuah referensi ke Data File. Tapi bila file tak ada, atau bila CodeBase mendeteksi adanya error yang lain, sebuah pesan error akan ditampilkan dan NULL pointer dikembalikan.

Yang perlu diperhatikan sebelum mengakhiri program, adalah menutup seluruh Data File yang telah dibuka sebelumnya. Ini memastikan bahwa seluruh perubahan pada Data File akan di-update dengan benar.

Ada dua Function yang biasa dipakai, pertama **d4close** dan kedua adalah **d4close_all**. Function **d4close** menutup Data File yang disebutkan dengan parameter yang ada di pointer **DATA4**. Function **d4close_all** menutup seluruh data, index, dan memo file yang telah dibuka.

2.8.4 Akses Ke Field

Untuk bisa akses field Data File terebut, hal pertama yang harus dilakukan adalah memberikan pointernya ke struktur **FIELD4**. Untuk ini ada 2 cara, pertama memberitahukan pada nomor posisi field keberapa, field yang akan diakses itu. Kedua adalah menginputkan nama field yang akan diakses, tentu saja ini berarti harus tahu lebih dahulu nama field tersebut.

2.8.4.1 Dengan Input Nomor Field

Tiap field di Data File mempunyai nomor field yang unik (berbeda satu dengan yang lain). Penomoran Field dari satu hingga jumlah maksimum field dalam

Data File tersebut. Jadi nomor ini juga menentukan posisi dalam record di Data File. Function **d4field_j** menggunakan nomor field untuk memberikan pointer kepada field struktur **FILED4**.

Maka untuk pertama dicari dulu jumlah maksimum field-nya. Kemudian dengan sebuh 'for' di looping untuk mengeluarkan tiap field-nya melalui pointer **FIELD4**.

```
Num_Field = d4num_field(Data_File);

for(j=1; j<Num_Field; j++)
{
        Field_Ref = d4field_j(Data_File, j);
        .....
}
```

2.8.4.2 Dengan Nama Field

Ini hanya bisa dilakukan bila nama itu, yang akan dicari field-nya sudah diketahui. Caranya dengan digunakannya field dari pointer **FIELD4**.

Function **d4field** akan mengembalikan sebuah harga pointer ke struktur **FIELD4**, yang mana pointer itu menunjukkan pada nama yang diberikan.

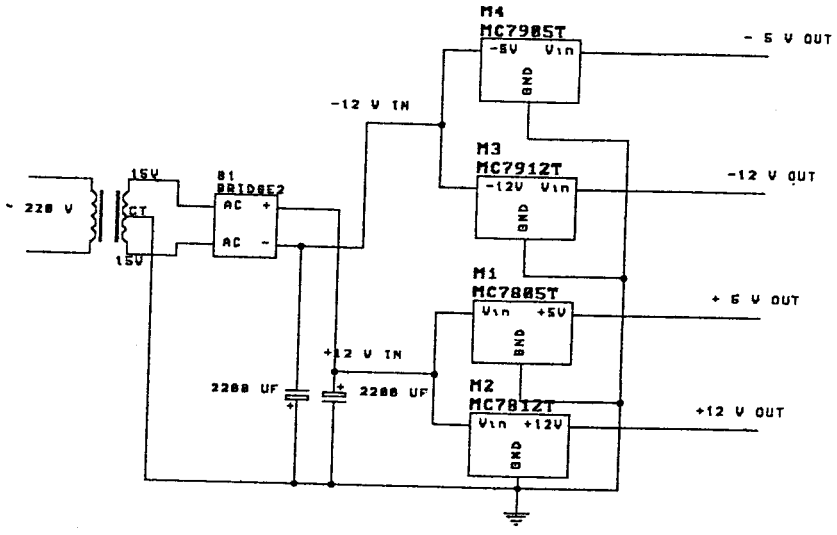
```
FILED4 *F_Nama, *L_Nama, *Alamat, *Umur, *Tgl_Lhr, *Kawin,
*Jumlah, *Comment;

//..... Data File Dibuka .....
        F_Nama = d4field(Data_File, "F_NAMA");
        L_Nama = d4field(Data_File, "L_NAMA");
        Alamat = d4field(Data_File, "ALAMAT");
        Umur = d4field(Data_File, "UMUR");
        Tgl_Lhr = d4field(Data_File, "TGL_LHR");
        Kawin = d4field(Data_File, "KAWIN");
        Jumlah = d4field(Data_File, "JUMLAH");
        Comment = d4field(Data_File, "COMMENT");
```

BAB III
PERENCANAAN
DAN PEMBUATAN

3.1 PERENCANAAN PERANGKAT KERAS

3.1.1 Unit Power Supply

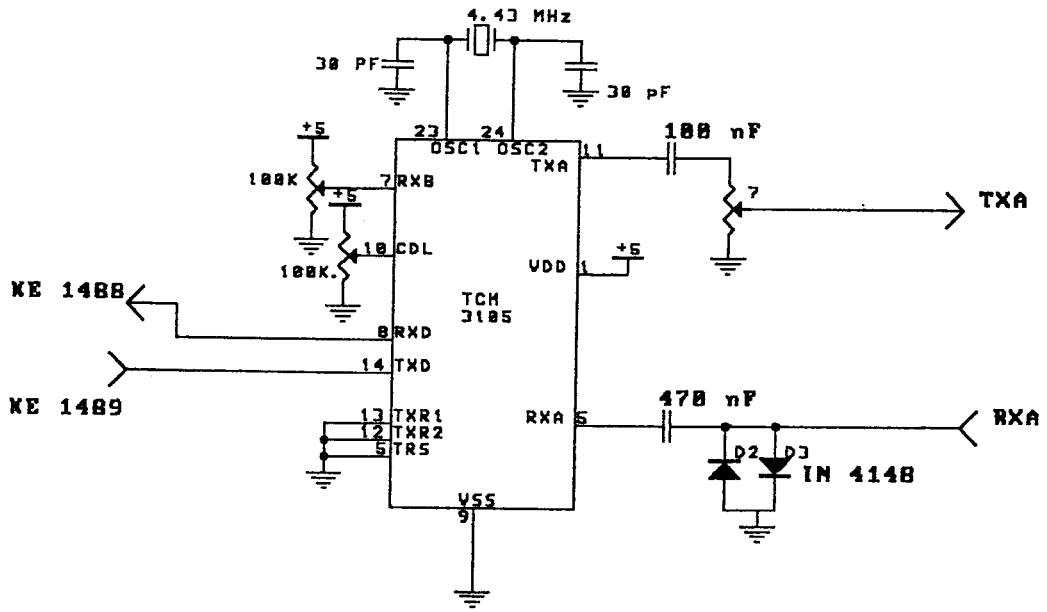
Merupakan Unit yang berfungsi untuk mensuplai daya yang dipakai oleh semua unit yang lain. Dengan cara merubah tegangan AC 220 V menjadi tegangan dc 5 Volt 4,5 Ampere, 24 volt 0,75 Ampere, 12 Volt dan -12 Volt yang masing-masing 1,25 Ampere.



Gambar 3-1 : Rangkaian Unit Power Supply

3.1.2 Unit Modem

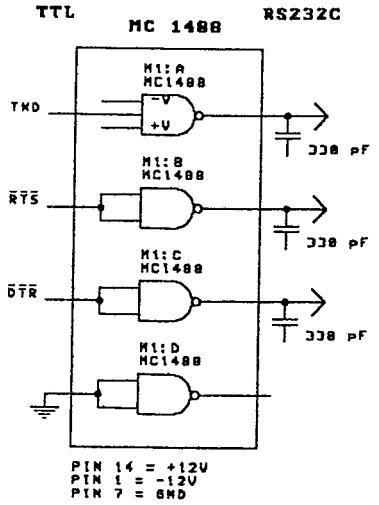
Fungsi dari unit ini adalah untuk komunikasi dari Terminal ke Stasiun Kontrol atau sebaliknya. Data yang akan dikirim, sebelumnya dirubah dulu dari bentuk digital ke sinyal analog dengan dua buah frekwensi yang berbeda, yaitu untuk Logic High dan Logic Low. Untuk selanjutnya data Analog tersebut dikirim lewat saluran komunikasi. Jika data analog tersebut sampai ke tempat tujuan maka data tersebut kembali diubah menjadi bentuk digital.



Gambar 3-2 : Rancangan Rangkaian Modem

3.1.3 Unit Current Loop Communication

Fungsi unit ini adalah untuk komunikasi dari Stasiun Sever Kontrol ke Terminal Kontrol atau sebaliknya. Unit ini juga menggunakan RS 232C tetapi tidak menggunakan modem melainkan dengan arus.

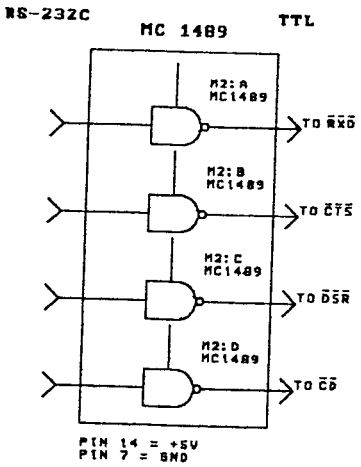


Gambar 3-3 : Rancangan untuk IC 1488

Cara kerja rangkaian ini adalah jika ada komunikasi data dari Terminal Kontrol ke Server Kontrol atau sebaliknya, maka data yang akan dikirim diubah levelnya yaitu:

1. Untuk Level High (5V) diubah menjadi Sumber arus 20 mA.
2. Untuk level Low (0V) diubah menjadi sumber arus 0 mA.

Setelah data dikirim, diterima kemudian kembali diubah menjadi seperti semula yaitu Level High adalah 5V dan level Low adalah 0 Volt.



Gambar 3-4 : Rancangan untuk IC 1489

3.2 PEMBUATAN PERANGKAT LUNAK

Perangkat lunak dibuat dengan bahasa assembly untuk kemudian dijalankan pada PC, yang akan mengendalikan jalannya peralatan ini secara keseluruhan, dimana secara umum fungsi software ini adalah :

1. Inisialisasi dan akses Serial COM.
2. Inisialisasi dan akses ke Data Base File.

3.2.1 Software Untuk Serial COM

Langkah pertama adalah menginisialisasi Card Serial (COM1 atau COM2). Ini diperlukan untuk membaca / mengirimkan data yang dikirim / diterima oleh Terminal Control melalui Serial COM1 atau COM2. Satu hal yang perlu diperhatikan dalam inisialisasi Card Serial adalah struktur bit-bit yang akan dipakai untuk pemrogaman Serial sebagai berikut :

1. Line Control Register

Menampung ketetntuan yang dipilah untuk menentukan berapa jumlah bit bagi setiap data, berapa jumlah stop bitnya, apakah menggunakan parity check, Disamping itu juga melayani apakah akan menentukan / mengubah baud rate divisor. Rincian bit-nya adalah :

Bit 7 : 1 = Menyatakan bahwa baud rate divisor akan ditentukan atau dirubah.

0 = Menyatakan bahwa baud rate divisor tidak diakses.

Bit 6 : 1 = Set break control aktif

Bit 5 : 1 = Parity bit ikut dikirimkan (stick parity)

Bit 4 : 1 = Dipilih parity genap

0 = Dipilih parity ganjil

Bit 3 : 1 = Parity check Enable

0 = Parity check Disable

Bit 2 : 1 = 2 Stop bit untuk 6 hingga 8 bit data, 1 1/2 Stop bit untuk 5 bit data

0 = 1 Stop bit

Bit 1	Bit 0	
0	0	= 5 bit data
0	1	= 6 bit data
1	0	= 7 bit data
1	1	= 8 bit data

2. Modem Control Register

Menampung pemrograman untuk mengatur modem, terutama pengguanan saluran DTR dan saluran RTS dari INS 8250 ke perangkat modem. Rincian bitnya adalah :

Bit 7 - Bit 5 : Harus selalu diberi 0

Bit 4 : 1 = Bila loopback internal diaktifkan, artinya data yang dikirim akan diterima sendiri.

0 = Bila loopback internal tidak diaktifkan, artinya data yang dikirim tidak diterima sendiri.

Bit 3 & 2 : Saluran penghubung ke perangkat lain, dapat diberi "1" atau "0"

Bit 1 : 1 = Berarti saluran RTS diaktiifkan

0 = Berarti saluran RTS tidak diaktifkan (normal)

Bit 0 : 1 = Berarti saluran DTR diaktifkan

0 = Berarti saluran DTR tidak diaktifkan (normal)

3. Line Status Register

Menampung bit-bit yang menyatakan keadaan kehadiran data dan keadaan kesalahan operasi. Rincian ketentuan bit-bit yang menampung keadaan (status) itu adalah sebagai berikut :

Bit 7 : harus selalu diberi 0

Bit 6 : 1 = Menyatakan bahwa Transmitter Shift Register sudah kosong

Bit 5 : 1 = Menyatakan bahwa isi Tx Register sudah kosong

Bit 4 : 1 = Terjadi Interupsi yang menghentikan (Break Interupt)

Bit 3 : 1 = Terjadi Framing Error (Jumlah bit pada setiap data tidak benar)

Bit 2 : 1 = Ada parity error

Bit 1 : 1 = Terjadi Overrun Error (Data yang masuk saling menimpa)

Bit 0 : 1 = Menyatakan bahwa sudah ada data yang mesuk di Rx Buffer (Data Ready)

4. Modem Status Register

Menampung bit-bit yang menyatakan keaadaan tentang hubungan dengan modem. Rincian bitnya adalah sebagai berikut :

Bit 7 : 1 = Menyatakan bahwa sinyal RLSD (Receive Line Signal Detect) sudah aktif

Bit 6 : 1 = Menyatakan bahwa Ring Indicator sudah aktif

Bit 5 : 1 = Menyatakan Data Set Ready (DSR) sudah dalam keadaan aktif.

Bit 4 : 1 = Menyatakan Clear to Send (CTS) sudah dalam keadaan aktif.

Bit 3 : 1 = Indikator perubahan Received Line Signal Detect

Bit 2 : 1 = Trailing Edge Ring Indicator

Bit 1 : 1 = Menyatakan ada perubahan keadaan di Data Set Ready (DSR berubah-ubah).

Bit 0 : 1 = Menyatakan ada perubahan di saluran Clear To Send (keadaan di CTS berubah-ubah),

Sehingga program inisialisasi dibuat sebagai berikut :

```
Tx_Buff1          equ   3f8H
Rx_Buff1          equ   3f8H
Baud_Div_L1 equ   3f8H
Baud_Div_M1 equ   3f9H
Int_Enb_R1  equ   3f9H
Int_Id_R1         equ   3faH
Line_Con_R1 equ   3fbH
Mod_Con_r1  equ   3fcH
Line_Stt_R1 equ   3fdH
Mod_Stt_R1  equ   3feH
```

3.2.2 Untuk Akses Data Base File

3.2.2.1 Inisialisasi, Open, Dan Close Data File

Bagian ini awal dari proses pada Data File. Terutama Inisialisasi program terhadap Codebase. Bila sukses Codebase akan mengembalikan harga ke pointer Ptr_Cb.

```
d4init(&Ptr_Cb);
        Ptr_Cb.auto_open= 0;
        Ptr_Cb.open_error= 0;
        Ptr_Cb.safety= 0;
Open();
.....
Close();
```

Disini terlihat ada rutin Open() dan Close(), yaitu untuk membuka Data File, dan Field-Field-nya. Dimana ini hanya dapat dilakukan bila inisialisasi sukses. Setelah proses terhadap Data File selesai, haruslah diakhiri dengan menutup Data File, dengan rutin Close().

Pada rutin Open() proses inisialisasi dilanjutkan pada taraf Field-Field dari Data File, dan Index-nya. Kesemuanya mengembalikan harga pointer pada P_Nrp, untuk Field Nrp dari Data.dbf, dan juga pointer P_Valid, P_Nm, dan P_Kd, untuk Field Valid, Nama, dan Kode dari Data.dbf.

```
void Open(void)
{       Ptr_Fl= d4open(&Ptr_Cb, "DATA");
        if(Ptr_Fl==NULL)
                Ptr_Fl= d4create(&Ptr_Cb,"DATA.DBF" ,F_INFO,NULL);
        if (Ptr_Fl != NULL)
        {P_Index= i4create(Ptr_Fl, "DATA", T_INFO);

         P_Nrp= d4field(Ptr_Fl,"NRP");  P_Valid= d4field(Ptr_Fl,"VALID");
         P_Nm= d4field(Ptr_Fl,"NAME");  P_Kd= d4field(Ptr_Fl,"KODE");

         P_Tag= d4tag(Ptr_Fl, "NRP_TAG");  d4tag_select(Ptr_Fl, P_Tag);
        }
```

Seperti di atas, juga terjadi hal yang sama pada Data File Data1.dbf, yang akan mengisi pointer untuk Field Nrp1, Name1, Kode1, Date, dan Jam.

```
.....
 P_Index= i4create(Ptr_Fl1, "DATA1", T_INFO1);

 P_Nrp1= d4field(Ptr_Fl1, "NRP1");  P_Nm1= d4field(Ptr_Fl1, "NAME1");
        P_Date= d4field(Ptr_Fl1,"DATE");  P_Jam= d4field(Ptr_Fl1, "JAM");
        P_Kd1= d4field(Ptr_Fl1, "KODE1");

 P_Tag1= d4tag(Ptr_Fl1, "NRP_TAG1");  d4tag_select(Ptr_Fl1, P_Tag1);
       }
}
```

Pada akhir program diharuskan untuk menutup Data File. Yaitu dengan rutin Close() sebagai berikut

```
void close()
{       d4close_all(&Ptr_Cb);  s4init_undo(&Ptr_Cb);
        mem4reset();  e4exit(&Ptr_Cb);
}
```

3.2.2.2 Edit, Tambah, Dan Hapus Data

Untuk proses Edit (atau perubahan data), tambah data, dan hapus data, diperlukan rutin-rutin seperti berikut

```
void Edit_rec(void)
{       Search();
        .....
        f4assign(P_Nrp, Nrp); f4assign(P_Valid, Valid);
        f4assign(P_Nm, Name); f4assign(P_Kd, Kode);
}
```

Untuk proses Edit, dan hapus, dilakukan setelah pencarian data dengan rutin Search(). Setelah pointer menunjuk pada Record yang benar, maka proses Edit dilakukan dengan mengganti isi Field-Fieldnya, dengan f4assign().

Agak berbeda pada proses tambah data, rutin diawali d4append_start(), dan diakhiri dengan d4append().

```
void Tambahrec(void)
{       .....
        d4append_start(Ptr_Fl, 0);
         f4assign(P_Nrp, Nrp); f4assign(P_Valid, Valid);
         f4assign(P_Nm, Name); f4assign(P_Kd, Kode);
         d4append(Ptr_Fl);
}
```

Sedang untuk proses hapus, setelah proses Search(), dilihat apakah data ditemukan dalam Data File. Bila ditemukan, pointer diarahkan pada lokasi tersebut dengan d4go(), selanjutnya Record ditandai untuk penghapusan dengan d4delete(). Baru kemudian Record yang sudah ditandai dihapus dari Data File dengan perintah d4pack().

```
void Hapusrec(void)
{  Ok=1;
        .....
        Search();
        if (!strcmp(Temp, Nrp))
        {
         d4go(Ptr_Fl, Ptr); d4delete(Ptr_Fl);  d4pack(Ptr_Fl);
        }
```

```
        while (Ok)
        { if (strcmp(Temp, Nrp))
          { d4skip(Ptr_Fl, 1L); u4ncpy(Temp, f4str(P_Nrp), 10);
                if (Ptr == Max)
                { gotoxy(23, 19);cprintf(" TIDAK Ditemukan ! ");
                }
                else Ptr++;
          }
          else { d4go(Ptr_Fl, Ptr); d4delete(Ptr_Fl); d4pack(Ptr_Fl);
                 Ok= 0;
                }
        }
   }
   else { textcolor(0);
                gotoxy(23, 19);cprintf(" TIDAK Ditemukan ! ");
        }
}
```

3.2.2.3 Search Record

Untuk mencari data Nrp yang sama dengan data hasil input dari Terminal Control, pada Data File, diperlukan rutin Search() ini. Pertama Data File dibaca dengan menggunakan Index yang ada, kemudian Record pertama dibaca dan dibandingkan dengan data hasil Input-an. Bila salah, pointer di-Increment untuk membaca Record selanjutnya.

```
void Search()
{ P_Tag= d4tag(Ptr_Fl, "NRP_TAG");
  d4tag_select(Ptr_Fl, P_Tag);
  if (d4seek(Ptr_Fl, Nrp) == 0)
  { Ptr= d4recno(Ptr_Fl);
        Max= d4reccount(Ptr_Fl);
        u4ncpy(Temp, f4str(P_Nrp), sizeof(Temp));
        if (!strcmp(Temp, Nrp))
        .....
```

BAB IV
CARA PEMAKAIAN ALAT

4.1 Persiapan

1. Pasang kabel serial yang menghubungkan Modem dengan RS-232.

2. Pasang kabel Line yang menghubungkan antara Modem di Server Kontroler dengan Modem di Terminal Kontrol.

3. Kemudian hidupkan kedua Modem.

4.2 Di Terminal Kontrol

1. Komputer Terminal Kontrol juga dihidupkan dan dijalankan program "TORX.EXE".

Pada saat itu seharusnya program "ROUT.EXE" sudah dijalan di Server Kontrol.

3. Bila sudah terhubungi akan tertulis "Koneksi ke Server Kontrol sudah terjalin. Siap membaca kartu." maka selesai.

4. Bila tertulis "Server tidak bisa dihubungi !".

5. Setelah komunikasi terjalin, di Terminal Kontrol terbaca "Masukkan Kartu Pengenal Anda". Masukkan kartu identitas ke Barcode reader.

6. Setelah data dibaca oleh Barcode, akan tampil "Data Valid" bila terdapat di Data Base. Bila tidak ada atau tidak sesuai dengan kode di Nomer di Data Base akan tampil "Data Tidak Valid".

4.3 Program Route.Exe di Server Kontrol

1. Jalankan program "ROUT.EXE" di Server, sehingga akan tampil Jam dan Tanggal saat itu.

2. Bila data yang diterima Modem Valid, akan tampil pula tulisan "Data Valid". Bila tidak akan tampil "Data Tidak Valid".

4.4 Program Svr.Exe di Server Kontrol

4.4.1 Penambahan Data Base

1. Pilih menu 'Master', hingga keluar window daftar Nama , Nomer dan Valid tidaknya orang tersebut.

2. Tekan tombol Insert , hingga tampil window Input. Kemudian isikan data orang yang akan ditambahkan dan tiap kali pengisian mengakhirinya dengan menekan tombol Enter.

3. Pengisiannya yaitu kolom 'Nomor' diisi nomor kode Barcode-nya. Pada kolom 'Nama' diisi namanya dengan panjang maksimum 25 karakter.

4. Bila ternyata Nomer kode Barcode-nya sama dengan salah satu Nomer yang ada di Data Base, maka program akan meminta isi ulang.

4.4.2 Penghapusan

1. Seperti pada penambahan Data Base di atas, pilih pula Menu Master. Tetapi kemudian tombol Delete.

2. Setelah tampil window untuk menghapus isikan Nomer kode Barcode yang akan dihapus dan diakhiri dengan menekan tombol Enter.

4.4.3 Pengeditan

1. Bila ingin mengedit atau merubah isi Data Base Master, gerakkan kursor dengan menekan tombol Up dan Down.

2. Setelah bar menunjukkan pada data yang dimaksud tekan Enter.

3. Setiap kali merubah isinya, baik field Nomer, Nama, maupun field Valid, akhiri dengan menekan tombol Enter.

4.4.4 Melihat Report

1. Bila ingin mengetahui isi Report karyawan yang berhasil meng-akses pintu, pilih menu Report dari menu utama.

2. Setelah tampil window menu Report, akan terlihat nama karyawan yang berhasil mengakses pintu, beserta jam dan tanggal masuknya.

3. Bila ingin melihat yang lain gerakkan bar ke atas dan ke bawah dengan menekan tombol Up dan Down.

4.4.5 Keluar (ESC)

1. Bila ingin keluar dari program dapat dengan menekan terus tombol kiri. Kemudian tekan tombol Enter 2 X.

2. Cara lain adalah menekan tombol ESC 2 atau 3X sampai keluar tiulisan EXIT di layar kiri atas, kemudian tekan tombol Enter.

BAB V
PENUTUP

5.1 Kesimpulan

• Dengan menggunakan Codebase hasil Code-nya, atau dengan kata lain program Exe hasilnya, menjadi sangat kecil (hanya 100 K -an).

• Dengan program Exe yang kecil, selain menghemat tempat, juga eksekusi program akan lebih cepat.

• Saat masuk karyawan, baik jam maupun tanggal, setiap saat dapat diketahui.

• Dengan dipakainya Modem untuk pengiriman dan penerimaan data digital, saluran untuk mengirim yang dipakai dapat berupa H.T. (Handy Talky) atau pemancar radio lainnya, dan juga telepon biasa.

• Walaupun tidak memakai alat-alat komunikasi tersebut, bisa juga antara modem dihubungkan langsung dengan kabel.

• Bila sedang tidak dipakai, perangkat *modem* di Server Controller, dapat dipakai sebagai modem biasa dengan Baud 1200.

5.2. Saran

• Yang perlu diperhatikan pertama kali adalah sudah terjalin komunikasi atau tidaknya diantara Stasiun Pengontrol, yang berfungsi sebagai pengendali, penyimpan dan pemroses data, dengan Terminal Kontrol, yang berfungsi sebagai input kode dan pengendali kunci pintu.

• Bila teknik pengiriman data ditingkatkan dengan cara mengacak data sebelum dikirim, akan lebih meningkatkan segi sekuritasnya, dengan kata lain lebih sulit dibajak.

• Pada perangkat FSK Modem hal-hal yang perlu diperhatikan adalah setting tegangan pada pin-pin TXA, RXB, dan CDL. Karena pin-pin inilah yang memegang peranan penting dalam menentukan unjuk kerja modem tersebut.

DAFTAR PUSTAKA

1. Borland, ***Borland C++ 3.1 Programmer Guide***, Borland International.

2. Borland, ***Borland C++ 3.1 Library Reference***, Borland International.

3. Campbell, Joe, ***C Programmer's Guide to Serial Communications***, Sams Publishing, 1994.

4. Douglas V. Hall, ***Microprosessor And Interfacing: Programming & Hardware***, McGraw-Hill, 1992

5. Grady, M. Tim, ***C! Programming Principles & Practices***, McGraw-Hill, 1989

6. Peter Norton & John Socha, ***Assembly Language Book for The IBM PC***, Brady Publishing, 1989.

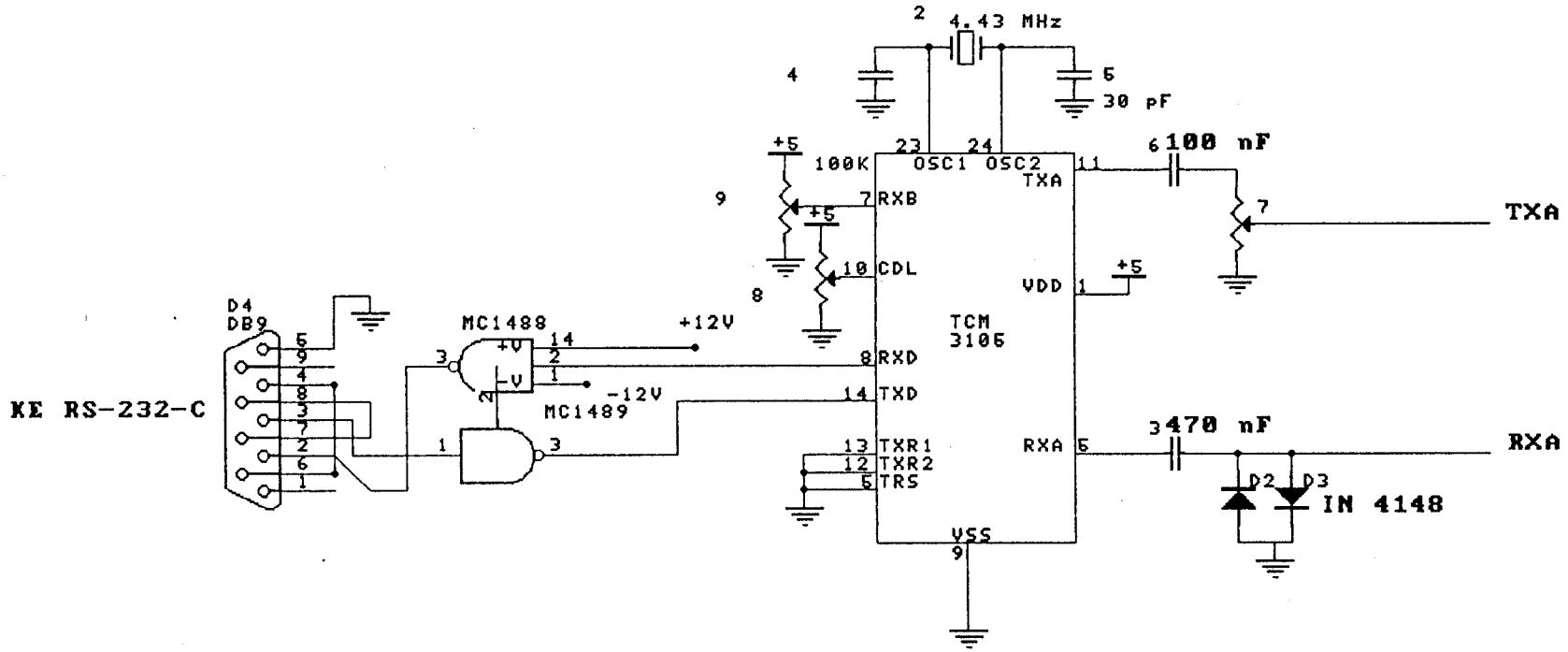
7. Robert L.Kruse-Bruce P. Leung-Clovis L.Tondo, ***Data Structures & Program Design In C***, Prentice Hall Int.

8. Sequiter, ***CodeBase 5.0 Reference Guide***, Sequiter Software Inc.

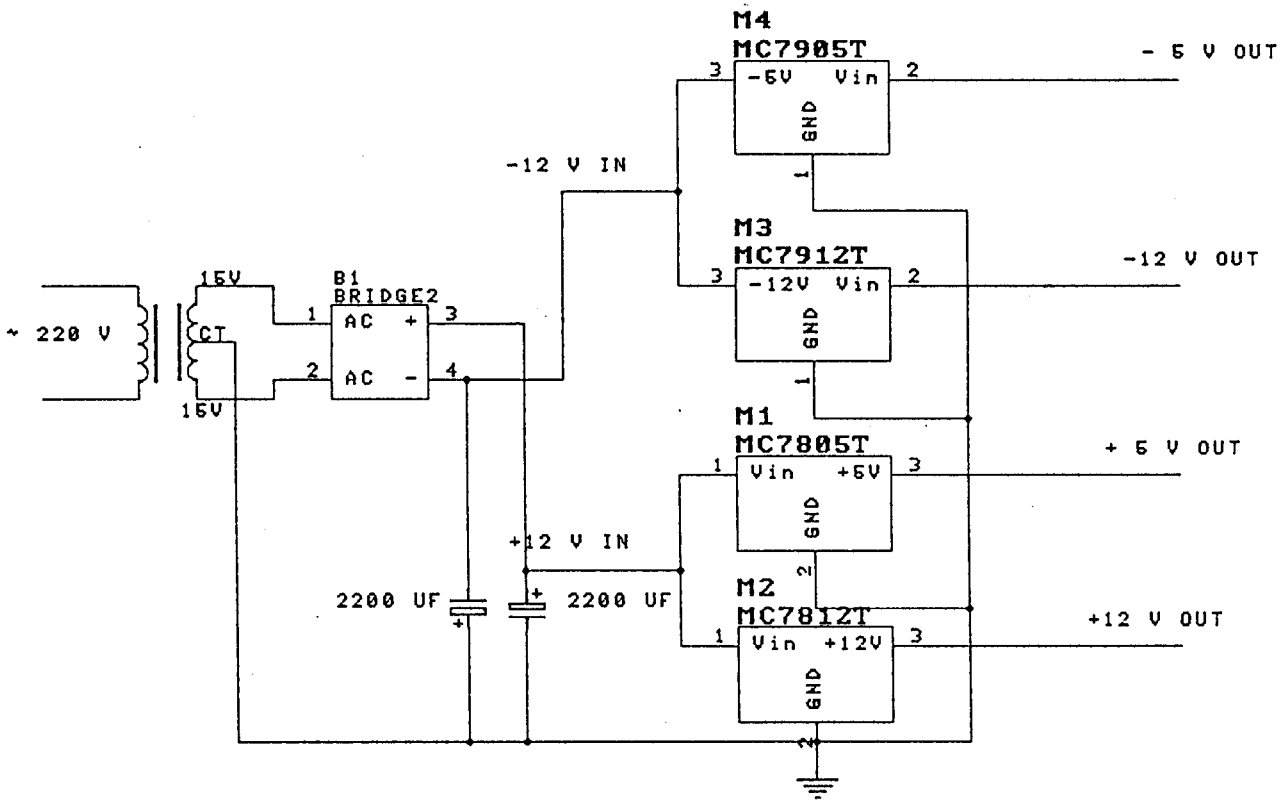
9. Sequiter, ***CodeBase 5.0 User's Guide***, Sequiter Software Inc.

10. Texas Ins., ***Telecomunication Circuit Data Book***, Texas Instrument Inc., 1993.

11. Wasito S., ***Data Sheet Book (Kumpulan Data Penting Komponen Elektronika)***, PT Elex Media Komputindo, 1992.

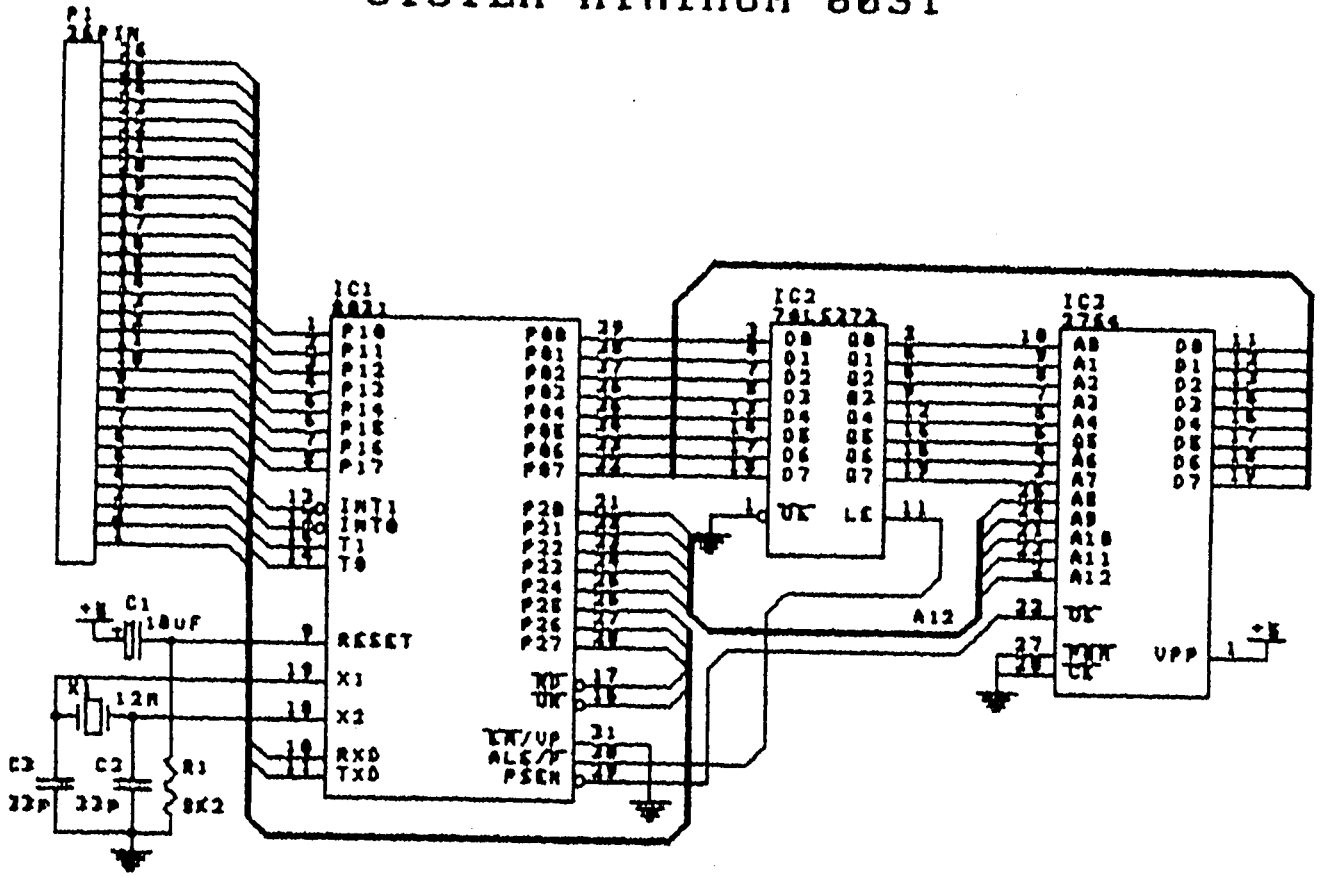


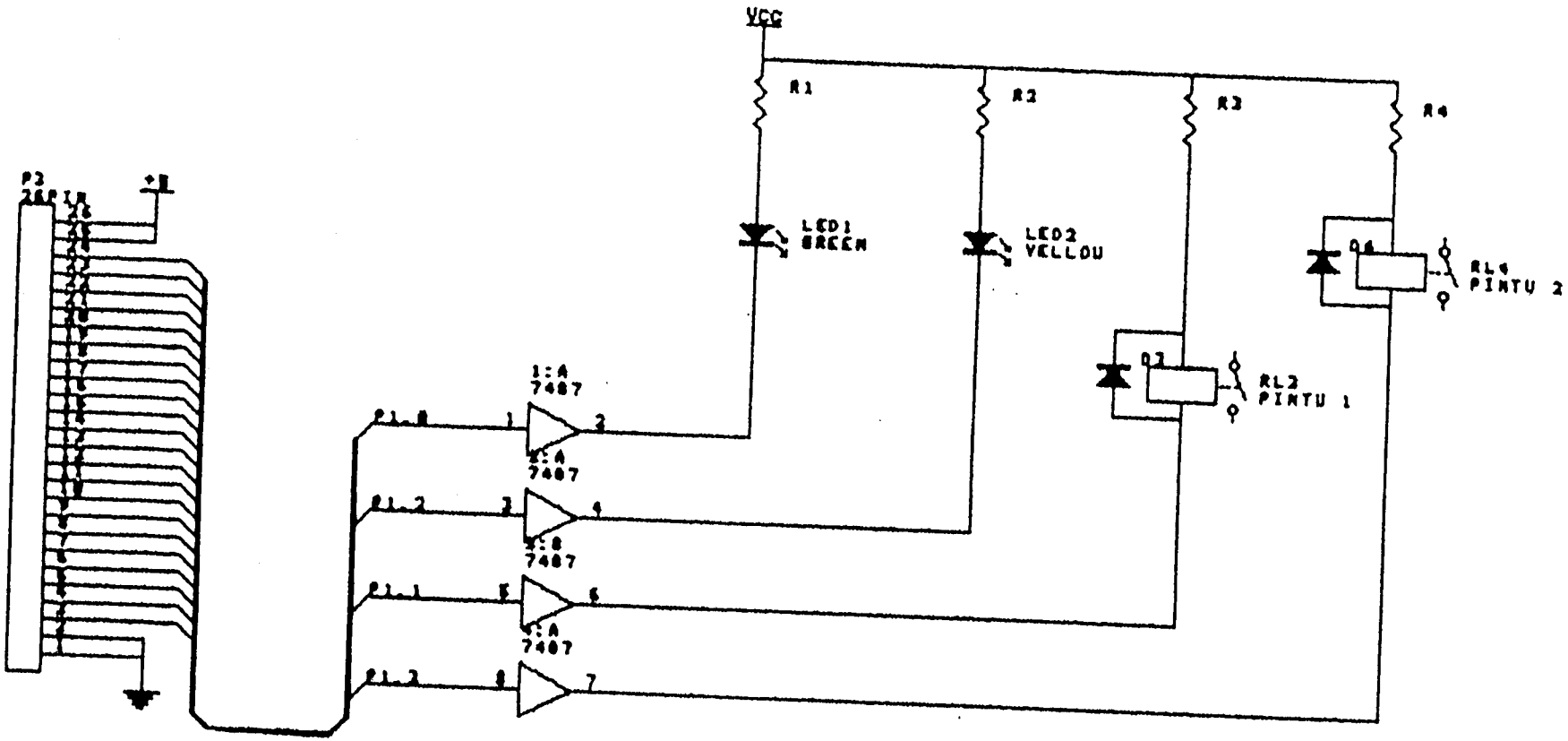
Title: RANGKAIAN LENGKAP MODEM 3105		
Size: A4	Number	Revision
Date: 1-AUG 1996		Sheet of
File: MOD-LENG/1		Drawn By: MBENK



Title: RANGKAIAN LENGKAP POWER SUPPLY		
Size: A4	Number:	Revision:
Date: 1-AUG 1995		Sheet of
File: PS-LENG/1		Drawn By: MBENK

SISTEM MINIMUM 8031




VCC
R1
R2
R3
R4
LED1
BRCEM
LED2
VELLOU
Q4
RL4
PINTU 2
Q3
RL3
PINTU 1
1:A
7487
2:A
7487
3:B
7487
4:A
7487
P3
P1.0
P1.2
P1.1
P1.3
1
2
3
4
5
6
8
7

```
; EPROM 8031 Assembly Program
; 4 TA
; By        : Atmojo Softw.are
; Last Modf : September 1995
;

KBTemp EQU 43H
CrTemp   EQU 44H
TEMP     EQU      87H

;=============================================================
; OUT PUT To Open The D[illegible]
; KEYBOARD 1 : P1.4 = DATA  P1.5 = CLOCK  P3.2 = ENCLCK
; KEYBOARD 2 : P1.6 = DATA  P1.7 = CLOCK  P3.3 = ENCLCK
; READY INDICATOR = T0 (P3.4), NO CONECT = T1 (P3.5)
;=============================================================

         org 0000h
         MOV      R3,00H
         DJNZ     R3,$
         ajmp     START

         ORG      40H
START:   ACALL    SP_INIT
         SETB     P3.2
         SETB     P3.3
         MOV      P1,#0FFH
KIRIM:
         MOV      R4,#0FFH
         DJNZ     R4,$
         MOV      R3,#7FH
R_RCV1:  CLR      P1.0                ;P1.0 = Warning / Error / Nothing
         MOV      A,#'S'
         ACALL    TRANSMT
         ACALL    RECEIVE
         MOV      A,#'S'     ;00H
         CJNE     A,CrTemp,ULGRCV
         AJMP     RCVDAT
ULGRCV:  DJNZ     R3,R_RCV1
;    SETB T1                 ; NO CONECTION
         AJMP     $          ;KIRIM
RCVDAT:  SETB     P1.0
;        SETB     T0       ; READY
TESKB:   CLR      P1.2                ;P1.2 = READY LED
         JNB      P1.4,R_TES2
         JNB      P1.6,TESKY1
         SJMP     TESKB

R_TES2:  ACALL    INKEY
LNJKY2:
         ;MOV     SBUF,#'A'

         MOV      A,#0DH
         CJNE     A,CrTemp,WR_KB

         ;MOV     SBUF,#'A'

         MOV      A,#'A'
         ACALL    TRANSMT

         ;MOV     SBUF,#0DH

         MOV      A,#0DH
         ACALL    TRANSMT
         AJMP     RCV_C

WR_KB:   MOV      A,CrTemp
         ACALL    TRANSMT
```

```
            AJMP    R_TES2
TESKY1:     ACALL   INKY1
LNJKY3:
            ;MOV    SBUF,#'B'

            MOV     A,#0DH
            CJNE    A,CrTemp,W_KB1

            ;MOV    SBUF,#'B'

            MOV     A,#'B'
            ACALL   TRANSMT

            ;MOV    SBUF,#0DH

            MOV     A,#0DH
            ACALL   TRANSMT
            AJMP    RCV_C

W_KB1:      MOV     A,CrTemp
            ACALL   TRANSMT
            AJMP    TESKY1
RCV_C:      ACALL   RECEIVE
            ANL     CrTemp,#0FH
            MOV     A,#00H
            CJNE    A,CrTemp,CASE1
            CLR     P1.0
            acall   dly1
            ACALL   DLY1
            SETB    P1.0
            AJMP    TESKB
CASE1:      MOV     A,#01H
            CJNE    A,CrTemp,CASE2
            CLR     P1.1                ;P1.1 = Door 1 /2(?)
            acall   dly1
            acall   dly1
            SETB    P1.1
            AJMP    TESKB
CASE2:      MOV     A,#02H
            CJNE    A,CrTemp,CASE3
            CLR     P1.3                ;P1.3 = Door 2 /1(?)
            ACALL   DLY1
            acall   dly1
            SETB    P1.3
            ajmp    TESKB
CASE3:      MOV     A,#03H
            CJNE    A,CrTemp,CASE4
            ACALL   DLY1
            AJMP    TESKB
CASE4:      MOV     A,#04H
            CJNE    A,CrTemp,end_C
            ACALL   DLY1
END_C:      NOP
;           mov     P1,#0FDH
;           ACALL   DLY1
            AJMP    TESKB


;INISIALISASI TIMER DAN SERIAL
SP_INIT:    MOV     TMOD,#00100000B
            MOV     TH1,#0E6H ;12M ,E8=11,059M ;BAUT RATE 1200
            MOV     TCON,#01000000B
            MOV     TEMP,#00H
            MOV     SCON,#01010010B     ;MODE 1,REN
            RET
```

```
RECEIVE: JNB RI,RECEIVE  ;TUNGGU SAMPAI SEMUA TERKIRIM
         CLR RI
         MOV A,SBUF      ;TERIMA DATA
         MOV     CrTemp,A
         RET

RCV:   JNB       RI,NODAT
       CLR       RI
       MOV       A,SBUF
       MOV       CrTemp,A
       RET

NODAT: MOV       CrTemp,#00H
       RET

TRANSMT: JNB     TI,$
       CLR       TI
       MOV       SBUF,A
       RET

DLY1:    MOV     R3,#07H
R_DLY:   MOV     R4,#0FFH
R_DLY1: MOV      R5,#0FFH
        DJNZ     R5,$
        DJNZ     R4,r_DLY1
        DJNZ     R3,R_DLY
        RET

K_BRD: MOV       R2,#08H
       MOV       A,#00H
       SETB      P3.2        ; CLOCK ENABLE
;      DJNZ      R2,$
       JB        P1.4,$ ; DATA
       JB        P1.5,$ ;NOKEY  ; CLOCK
       JNB       P1.5,$
R_KEY: JB        P1.5,$
       MOV       C,P1.4      ; DATA
       RLC       A
       JNB       P1.5,$
       DJNZ      R2,R_KEY
       JB        P1.5,$
       MOV       C,P1.4
       JNB       P1.5,$
       JB        P1.5,$
       JNB       P1.5,$
NOKEY: MOV       KBTemp,A
;      CLR       P3.2
       RET

INKEY:   SETB    P3.2
         MOV     CrTemp,#00H
R_INK: ACALL     K_BRD
       MOV       A,#00H
       CJNE      A,KBTemp,TSHF
;      CLR       P3.2
       RET

TSHF:  MOV       A,#48H
       CJNE      A,KBTemp,NXTES
       MOV       CrTemp,#50
       MOV       A,KBTemp
       AJMP      R_INK
NXTES:
       MOV       A,#0FH
       CJNE      A,KBTemp,R_INK
       ACALL     K_BRD
       MOV       A,#48H
       CJNE      A,KBTemp,CNV_
```

```
            MOV     CrTemp,#00H
            AJMP    R_INK
  CNV_:     MOV     DPTR,#BARCD
            MOV     R3,#00H
  R_SCN: INC        R3
            MOV     A,R3
            MOVC    A,@A+DPTR
            CJNE    A,BUFKEY,R_SCN
            MOV     A,R3 ;CrTemp
            MOV     DPTR,#PCASC
            MOVC    A,@A+DPTR
            MOV     CrTemp,A
  ;         CLR     P3.2
            RET

  K_BRB: MOV        R2,#08H
            MOV     A,#00H
            JB      P1.6,$      ; DATA
            JB      P1.7,$ ;NOKEY
            JNB     P1.7,$
  R_KY1: JB         P1.7,$
            MOV     C,P1.6      ; DATA
            RLC     A
            JNB     P1.7,$
            DJNZ    R2,R_KY1
            JB      P1.7,$
            MOV     C,P1.6
            JNB     P1.7,$
            JB      P1.7,$
            JNB     P1.7,$
  NOKY1: MOV        KBTemp,A
            RET

  INKY1:    MOV     CrTemp,#00H
            SETB    P3.3
  R_IK1: ACALL      K_BRB
            MOV     A,#00H
            CJNE    A,KBTemp,TSHF1
  ;         CLR     P3.3
            RET

  TSHF1: MOV        A,#48H
            CJNE    A,KBTemp,NXT1
            MOV     CrTemp,#50
            MOV     A,KBTemp
            AJMP    R_IK1
  NXT1:
            MOV     A,KBTemp
            MOV     A,#0FH
            CJNE    A,KBTemp,R_IK1
     ACALL          K_BRB
     MOV A,#48H
            CJNE    A,KBTemp,CNV_1
            MOV     CrTemp,#00H
     AJMPR_IK1
  CNV_1:    MOV     DPTR,#BARCD
     MOV R3,#00H
  RSCN1: INC        R3
            MOV     A,R3
     MOVC           A,@A+DPTR
     CJNE A,BUFKEY,RSCN1
            MOV     A,R3 ;CrTemp
            MOV     DPTR,#PCASC
     MOVC           A,@A+DPTR
            MOV     CrTemp,A
  ;         CLR     P3.3
            RET
```

```
BARCD:  DB 68H,78H,64H,0A4H,74H,6CH,0BCH,7CH,62H,0A2H
        DB 72H,0AAH,66H,0A8H,0B8H,24H,0B4H,34H,0ACH,3CH
        DB 0C2H,22H,0B2H,2AH,0DAH,38H,0D8H,0C4H,0D4H,2CH
        DB 0CCH,0DCH,42H,0D2H,32H,4AH,5AH,58H,44H,20H
        DB 64H,4CH,20H,5CH,20H,20H,052H,6EH,4AH,29H

PCASC:  DB 31H,32H,33H,34H,35H,36H,37H,38H,39H,30H
        DB 2DH,2BH,08H,51H,57H,45H,52H,54H,59H,55H
    DB 49H,4FH,50H,5BH,5DH,41H,53H,44H,46H,47H
    DB 48H,4AH,4BH,4CH,3AH,22H,0DH,5AH,58H,43H
    DB 56H,42H,76H,4DH,4DH,6DH,3FH,1BH,2FH,20H
    ;
    DB 76H,21H,40H,23H,24H,25H,5EH,26H,2AH,28H
    DB 29H,5FH,2BH,51H,1BH,51H,57H,56H,52H,54H
    DB 59H,55H,49H,4FH,50H,54H,5BH,0DH,41H,53H
    DB 44H,46H,47H,49H,4AH,4BH,4CH,33H,7CH,5AH
    DB 58H,43H,56H,42H,4EH,4DH,3CH,3EH,3FH,20H
        END


/*  Project: ROUT (Server to Terminal) 4 T.A.
                Name: ROUT.C
                Other Modul: C5FOX_M.LIB
                Mod. Memory: Medium

                Programmed by: Atmojo Software
                Last Modf. : September 1995
*/

#include <bios.h>
#include <conio.h>
#include <dos.h>
#include <stdio.h>
#include <process.h>
#include <string.h>
#include "D4ALL.h"

#define COM1    0
#define COM2    1
#define DATA_READY 0x100
#define TRUE    1
#define FALSE   0
//#define SETTINGS (_COM_1200|_COM_CHR7|_COM_STOP1|_COM_NOPARITY)
//#define SETTINGS (0x80|0x02|0x00|0x00)            //idem ?
#define SETTINGS (_COM_1200|_COM_CHR8|_COM_STOP1|_COM_EVENPARITY)
/* Serial 1200 bps, 8 bit data, even vanity, 1 stop bit */
#define SATU        1
#define DUA                 2
#define TIGA        3
#define EMPAT       4
#define LIMA        5
#define ENAM        6


#ifdef __TURBOC__
  extern unsigned _stklen = 10000;
#endif

CODE4  Ptr_Cb;
DATA4  *Ptr_Fl,*Ptr_Fl1;
FIELD4 *P_Nrp,*P_Valid,*P_Nm,*P_Kd,
                    *P_Nrp1,*P_Nm1,*P_Date,*P_Jam,*P_Kd1;
FIELD4INFO F_Info[]= {      {"NRP",r4str,25,0},
                            {"VALID",r4log,1,0},
                            {"NAME",r4str,25,0},
                            {"KODE",r4str,6,0},
                            {0,0,0,0},
                    };
```

```
FIELD4INFO F_Info1[]= { {"NRP1",r4str,25,0},
                        {"NAME1",r4str,25,0},
                        {"DATE",r4str,8,0},
                        {"JAM",r4str,8,0},
                        {"KODE1",r4str,6,0},
                        {0,0,0,0},
                      };

TAG4 *P_Tag, *P_Tag1;

char In_BC[20], C_Temp[2];
int rc, P_Masuk=0;

int Cek_Fl(char *Temp)
{  struct find_t File;
           int Ok=1;

           Ok= _dos_findfirst(Temp, _A_NORMAL, &File);
           if (Ok)
            {  gotoxy(18, 18);
                       textcolor(12);
                       textbackground(15);
                       cprintf(" TAK ADA FILE '%s' ! ", Temp);
                       return 1;
            }
           return 0;
}

void OpenDataFile(void)
{
           if (Cek_Fl("DATA.DBF"))
                       { getch();
                       exit(0); }
           else
           { Ptr_Fl= d4open(&Ptr_Cb, "data");
            e4exit_test(&Ptr_Cb);

            if (Ptr_Fl != NULL)
            { P_Nrp= d4field(Ptr_Fl, "NRP");
                       P_Valid= d4field(Ptr_Fl, "VALID");
                       P_Nm= d4field(Ptr_Fl, "NAME");
                       P_Kd= d4field(Ptr_Fl, "KODE");
            }
           }

           Ptr_Fl1= d4open(&Ptr_Cb, "data1");
           if (Ptr_Fl1 == NULL)
                       Ptr_Fl1= d4create(&Ptr_Cb, "DATA1", F_Info1, NULL);
           e4exit_test(&Ptr_Cb);

           if (Ptr_Fl1 != NULL)
           { P_Nrp1= d4field(Ptr_Fl1, "NRP1");
            P_Nm1= d4field(Ptr_Fl1, "NAME1");
            P_Date= d4field(Ptr_Fl1, "DATE");
            P_Jam= d4field(Ptr_Fl1, "JAM");
            P_Kd1= d4field(Ptr_Fl1, "KODE1");
           }
}

/*
void PrintRecords(void)
{
 int    rc,i,age_value;
           double  amount_value;
 char   f_name_str[15],l_name_str[15];
 char   address_str[20];
 char   date_str[9];
 char   married_str[2];
```

```
 for(rc = d4top(Ptr_Fl);rc == r4success
                                ;rc =  d4skip(Ptr_Fl, 1L))
 {
          f4ncpy(f_name,f_name_str,sizeof(f_name_str));
          f4ncpy(l_name,l_name_str
                                ,sizeof(l_name_str));
          f4ncpy(address,address_str
                                ,sizeof(address_str));
          age_value = f4int(age);
                amount_value = f4double(amount);
          f4ncpy(birth_date,date_str
                                        ,sizeof(date_str));
          f4ncpy(married,married_str,
                                sizeof(married_str));

          printf("-----------------------------\n");

          printf("Name    : %10s %10s\n",f_name_str
                                                ,l_name_str);
          printf("Address  : %15s\n",address_str);
          printf("Age : %3d  Married : %1s\n"
                                ,age_value,married_str);
          printf("Amount purchased this year: "
                        " $%5.2lf\n\n", amount_value);
 }
}
*/

void Tambah(char *f_nrp,char *f_name_str,char *f_jam,char *f_date,char *f_kode)
{
          d4append_start(Ptr_Fl1, 0);
          f4assign(P_Nrp1, f_nrp);
          f4assign(P_Nm1, f_name_str);
          f4assign(P_Date, f_date);
          f4assign(P_Jam, f_jam);
          f4assign(P_Kd1, f_kode);
          d4append(Ptr_Fl1);
}

int Cari(void)
{  char f_nrp[26], f_valid[2], S_Temp[20];
          int P_Str;
          u4ncpy(S_Temp, "\0", 20);
          P_Str= strlen(In_BC);
          sprintf(C_Temp, "%c", In_BC[0]);
          if (((strcmp(C_Temp, "S"))==0) || ((strcmp(C_Temp, "A"))==0) ||
                    ((strcmp(C_Temp, "B"))==0) ||
                    ((strcmp(C_Temp, "C"))==0))
          {
                    for (rc= 1; rc < P_Str; rc++)
                    {  sprintf(C_Temp, "%c", In_BC[rc]);
                              u4ncat(S_Temp, C_Temp, 20);
                    }
                    u4ncpy(In_BC, S_Temp, 20);
          }
          sprintf(C_Temp, "%c", In_BC[P_Str-1]);
          if (((strcmp(C_Temp, "S"))==0) || ((strcmp(C_Temp, "A"))==0) ||
                    ((strcmp(C_Temp, "B"))==0) ||
                    ((strcmp(C_Temp, "C"))==0))
          {
                    c4trim_n(S_Temp, (P_Str-1));
                    u4ncpy(In_BC, S_Temp, 20);
          }

          for(rc= d4top(Ptr_Fl); rc == r4success ;rc=  d4skip(Ptr_Fl, 1L))
          {
           f4ncpy(P_Nrp, f_nrp, sizeof(f_nrp));
```

```
            c4trim_n(f_nrp, sizeof(f_nrp));

                    if((strcmp(f_nrp, In_BC) == 0))
            {
                    f4ncpy(P_Valid, f_valid, sizeof(f_valid));
                    if((strcmp(f_valid, "T") == 0))  return 0;
                    else  return -1;
            }
           }
           return -1;
}

void Send(unsigned int Dt_S, unsigned int C_Mn)
{
           bioscom(_COM_SEND, Dt_S, C_Mn);
}

unsigned int Recv(unsigned int C_Mn)
{          unsigned int Data= 0;

           Data= bioscom(_COM_RECEIVE, 0, C_Mn);
           return Data;
}

void Init(unsigned int C_Mn)
{
           bioscom(_COM_INIT, SETTINGS, C_Mn);
}

void Buka_P(char P[7])
{
           c4trim_n(P, 7);

           if ((strcmp(P, "SATU")) == 0)
            if (P_Masuk == 1)
                                { Send(1, COM1);
                                        return;
                                }
                    //getch();
           if ((strcmp(P, "DUA")) == 0)
                    if (P_Masuk == 2)
                    { Send(2, COM1);
                                return;
                    }

           if ((strcmp(P, "TIGA")) == 0)
                    if (P_Masuk == 3)
                    {
                                Send(3, COM1);
                                return;
                    }
           if ((strcmp(P, "EMPAT")) == 0)
                    if (P_Masuk == 4)
                    {
                                Send(4, COM1);
                                return;
                    }
           if ((strcmp(P, "LIMA")) == 0)
           {        if (P_Masuk == 5)
                    {           Send(5, COM1);
                                return;
                    }

           }
           else
                    { printf("Pintu Masuk =\t S A L A H !!\n\n:");
                                Send(0, COM1);
                    }
```

```
}

void Gtime(char Temp2[9])
{  struct date d;

        getdate(&d);
        printf("The current year is: %d\n", d.da_year);
        sprintf(C_Temp, "%02d", d.da_year);
        u4ncpy(Temp2, C_Temp, 9);

        printf("The current day is: %d\n", d.da_day);
        sprintf(C_Temp, "%02d", (char) d.da_day);
        u4ncat(Temp2, C_Temp, 9);

        printf("The current month is: %d\n", d.da_mon);
        sprintf(C_Temp, "%02d", (char) d.da_mon);
        u4ncat(Temp2, C_Temp, 9);
}

void Ghour(char Temp2[9])
{  struct  time t;

        gettime(&t);
        printf("The current time is: %2d:%02d:%02d.%02d\n",
                        t.ti_hour, t.ti_min, t.ti_sec, t.ti_hund);
        sprintf(C_Temp, "%02d", (char)t.ti_hour);
                u4ncpy(Temp2, C_Temp, 9);
                u4ncat(Temp2, ":", 9);
        sprintf(C_Temp, "%02d", (char)t.ti_min);
                u4ncat(Temp2, C_Temp, 9);
                u4ncat(Temp2, ":", 9);
        sprintf(C_Temp, "%02d", (char)t.ti_sec);
                u4ncat(Temp2, C_Temp, 9);
}

void main1(void)
{ unsigned int Temp, Ok=1;
 char *t;
 int i;
 strcpy(In_BC, "\0");
 i=0;
 P_Masuk=0;
 while(Ok)
 {
        Temp=bioscom(_COM_STATUS, 0, COM1);
        if (Temp & DATA_READY)
        { if ((Temp=Recv(COM1)& 0x7F) !=0)
          {
                if( ((Temp==115) || (Temp==83)) )
                {
                        if ((Temp==97) || (Temp==65))
                        {       P_Masuk=SATU;
                                continue;
                        }
                        if ((Temp==98) || (Temp==66))
                        {       P_Masuk=DUA;
                                continue;
                        }
                        if (Temp !=13)
                        {
                         if ((Temp==99) || (Temp==67))
                         {  printf("\n:Data Terima %s \n:", In_BC);
                                 In_BC[i+1]= '\r';  In_BC[i+2]= '\n0';  return;
                         }

                         putch(Temp);
                                sprintf(t, "%c", Temp); strcat(In_BC, t); i++;
                        }
```

```
                    else if(Temp == 13)
                            { printf("\n:Data Terima %s \n:", In_BC);
                            //P_Masuk=0;
                                    return;
                            }
                }
                else
                        if((Temp==115) || (Temp==83))
                        { Send(Temp, COM1);
                         delay(10);

                         Temp= bioscom(_COM_STATUS, 0, COM1);
                         if (Temp & DATA_READY)
                         { if ((Temp= Recv(COM1)& 0x7F) != 0)
                                 {
                                  if( ((Temp==115) || (Temp==83)))                    //!((s)|(S))
                                  {
                                          printf("TERMINAL CONNECTION NOT ESTABLISH !\n\n:");
                                  }
                                  else
                                          { printf("TERMINAL CONNECTION ESTABLISH ..... \n\n:");
                                           if ((Temp= Recv(COM1)& 0x7F) != 0)
                                           {
                                                   putch(Temp);
                                                   if(Temp != 13)
                                                   { sprintf(t, "%c", Temp);
                                                    strcat(In_BC, t);
                                                    i++;
                                                   }
                                                   else if(Temp == 13)
                                                           { printf("\n:Data Terima %s \n:", In_BC);
                                                            return;
                                                           }
                                           }
                                          }
                                 }
                         }
                         else
                                 { printf("TERMINAL CONNECTION ESTABLISH ..... \n\n:");
                                  if ((Temp= Recv(COM1)& 0x7F) != 0)
                                  {
                                          putch(Temp);
                                          if(Temp != 13)
                                          { sprintf(t,"%c", Temp);
                                           strcat(In_BC, t);
                                           i++;
                                          }
                                          else if(Temp == 13)
                                                  { printf("\n:Data Terima %s \n:", In_BC);
                                                   return;
                                                  }
                                  }
                                 }
                        }
        }
        }

        if (kbhit())
        {
                if ((Temp = getch()) == '\x1B')                 // ==27 (ESC)
                        exit(0);

                Send(Temp, COM1); putch(Temp);
        }

}

}
```

```
void main(void)
{  //int rc;
          //char status[15];
  char f_nrp[26], f_valid[2], f_name[25], f_kode[7],
                    f_date[9], f_jam[9];

  d4init(&Ptr_Cb);
  Ptr_Cb.open_error=0;  Ptr_Cb.safety=0;  Ptr_Cb.auto_open=0;
  clrscr();  Gtime(f_date);  Ghour(f_jam);
  OpenDataFile();
  Init(COM1);
  printf("\n\n:");

  do { main1();
          rc=Cari();
          if(rc == 0)
          {
                    f4ncpy(P_Nrp, f_nrp, sizeof(f_nrp));
                    f4ncpy(P_Valid, f_valid, sizeof(f_valid));
                    f4ncpy(P_Nm, f_name, sizeof(f_name));
                    f4ncpy(P_Kd, f_kode, sizeof(f_kode));
                    Tambah(f_nrp, f_name, f_jam, f_date, f_kode);

                    printf("Yang Masuk =\t%s \n Pintu =\t%s \n\tData VALID \n\n:", f_name, f_kode);
                    Buka_P(f_kode);

                    /*Send('\n', COM1); putch('\n') ;
          }

          else
                    { //Send('\n', COM1);
                     printf("Yang Masuk =\tData TIDAK VALID !!\n\n:");
                     Send(0, COM1);
          }

          }while(!kbhit());

  d4close_all(&Ptr_Cb);
  d4init_undo(&Ptr_Cb) ;   /* free up memory */
  mem4reset() ;            /* free up memory */
}
```

CHAPTER 2

8051 Family Architecture

INTRODUCTION

The entire 8051 Family of 8-bit microcontrollers is based on the "core" architecture shown in Figure 2-1. The original member of this family is produced under the name 8051AH. The term "8051", however, is often used generically to refer to all of the 8051 Family members.

In this chapter the term "8052" is used to refer to an 8051AH with a double amount of ROM and RAM, and an extra timer called Timer 2. It is also included in this "core" discussion because its features are often found in other enhanced 8051 Family members. (see Members of the Family in Chapter 1).

The latter section of this data book details both the basic and enhanced 8051 Family members in separate chapters, but concentrates on the new features beyond the basic core architecture. Thus, the new reader should first concentrate on the features discussed in this chapter and the rest of Section I.

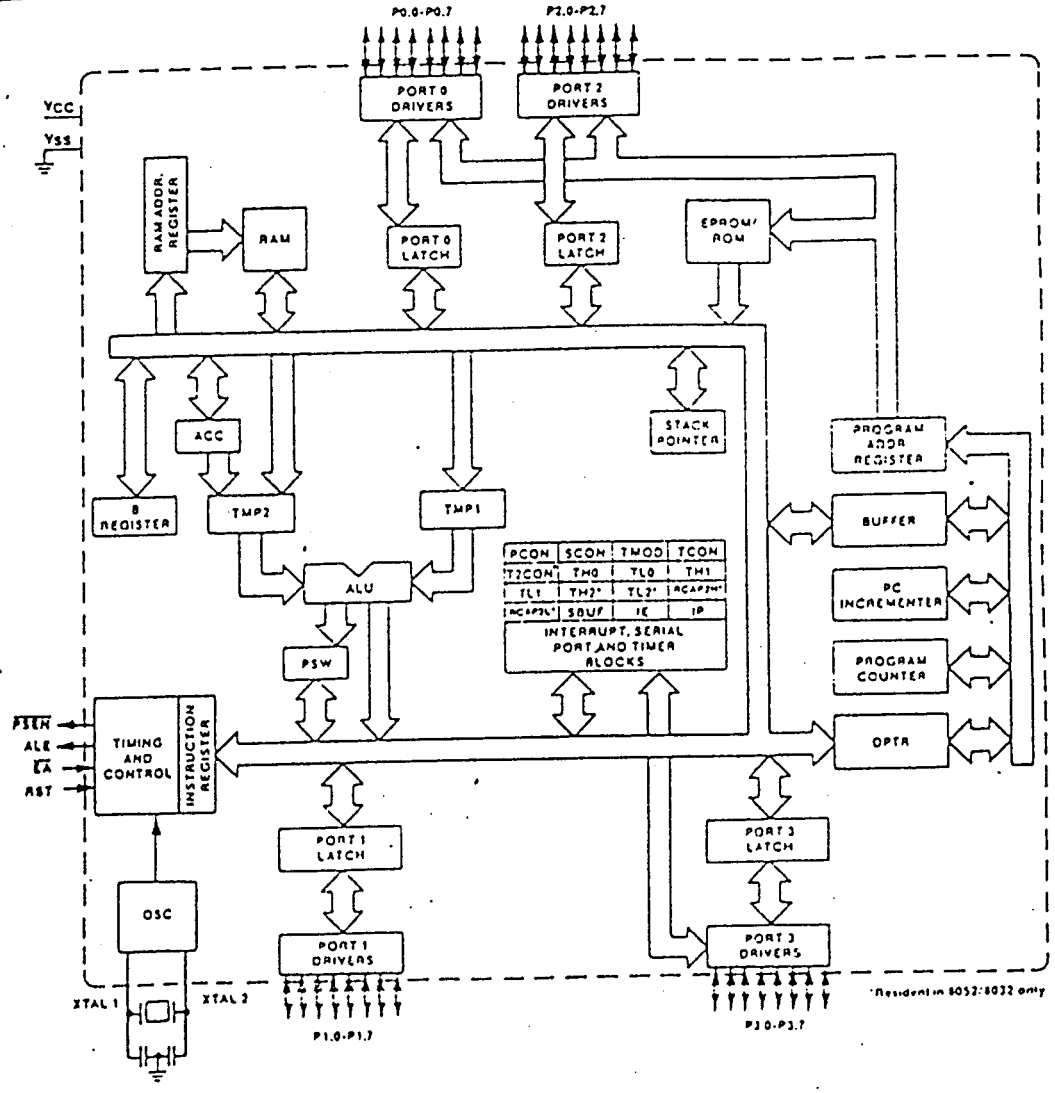


Figure 2-1. 8051 Family Architecture

Table 2-1 8051 Family Core Members

Part	Technology	On-Chip Program Memory (bytes)	On-Chip Data RAM (bytes)
8051AH	NMOS	4K ROM	128
8031AH	NMOS	-	128
8751H	NMOS	4K EPROM	128
8052	NMOS	8K ROM	256
80C51	CMOS	4K ROM	128
80C31	CMOS	-	128

The major 8051 Family features are:

- 8-Bit CPU
- On-Chip oscillator and clock circuitry
- 32 I/O lines
- 64K bytes address space for external Data Memory
- 64K bytes address space for external Program Memory
- Two 16-bit timer/counters (three on 8032/8052)
- A five-source interrupt structure (six sources on 8032/8052) with two priority levels
- Full duplex serial port
- Boolean Processor

MEMORY ORGANIZATION

The 8051 has separate address spaces for Program Memory and Data Memory. The Program Memory can be up to 64K bytes long. The lower 4K bytes (8K for 8052) may reside on-chip. The Data Memory can consist of up to 64K bytes of off-chip RAM, in addition to which it includes 128 bytes of on-chip RAM (256 bytes for the 8052), plus a number of "SFRs" (Special Function Registers) as listed below.

Symbol	Name	Address
*ACC	Accumulator	0E0H
*B	B Register	0F0H
*PSW	Program Status Word	0D0H
SP	Stack Pointer	81H
DPTR	Data Pointer (consisting of DPH and DPL	83H 82H
*P0	Port 0	80H
*P1	Port 1	90H
*P2	Port 2	0A0H
*P3	Port 3	0B0H
*IP	Interrupt Priority Control	0B8H
*IE	Interrupt Enable Control	0A8H
TMOD	Timer/Counter Mode Control	89H
*TCON	Timer/Counter Control	88H
+*T2CON	Timer/Counter 2 Control	0C8H
TH0	Timer/Counter 0 (high byte)	8CH
TL0	Timer/Counter 0 (low byte)	8AH
TH1	Timer/Counter 1 (high byte)	8DH
TL1	Timer/Counter 1 (low byte)	8BH
+TH2	Timer/Counter 2 (high byte)	0CDH
+TL2	Timer/Counter 2 (low byte)	0CCH
+RCAP2H	Timer/Counter 2 Capture Register (high byte)	0CBH
+RCAP2L	Timer/Counter 2 Capture Register (low byte)	0CAH
*SCON	Serial Control	98H
SBUF	Serial Data Buff	99H
PCON	Power Control	87H

The SFRs marked with an asterisk (*) are both bit- and byte-addressable. The SFRs marked with a plus sign (+) are present in the 8052 only. The functions of the SFRs are described as follows.

Accumulator

ACC is the Accumulator register. The mnemonics for accumulator-specific instructions, however, refer to the accumulator simply as A.

B Register

The B register is used during multiply and divide operations. For other instructions it can be treated as another scratch pad register.

Program Status Word

The PSW register contains program status information as detailed in Figure 2-2.

(MSB) (LSB)

CY	AC	F0	RS1	RS0	OV	—	P

Symbol	Position	Name and Significance
CY	PSW.7	Carry flag.
AC	PSW.6	Auxiliary Carry flag. (For BCD operations.)
F0	PSW.5	Flag 0 (Available to the user for general purposes.)
RS1	PSW.4	Register bank Select control bits 1 & 0. Set/cleared by software to determine working register bank (see Note).
RS0	PSW.3	
OV	PSW.2	Overflow flag.
—	PSW.1	(reserved)
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of "one" bits in the accumulator, i.e., even parity.

Note— the contents of (RS1, RS0) enable the working register banks as follows:

(0,0)—Bank.0 (00H-07H)
(0,1)—Bank 1 (08H-0FH)
(1,0)—Bank 2 (10H-17H)
(1,1)—Bank 3 (18H-1FH)

Figure 2-2. PSW: Program Status Word Register

Stack Pointer

The Stack Pointer register is 8 bits wide. It is incremented before data is stored during PUSH and CALL executions. While the stack may reside anywhere in on-chip RAM, the Stack Pointer is initialized to 07H after a reset. This causes the stack to begin at location 08H.

Data Pointer

The Data Pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). Its intended function is to hold a 16-bit address. It may be manipulated as a 16-bit register or as two independent 8-bit registers.

Ports 0 to 3

P0, P1, P2, and P3 are the SFR latches of Ports 0, 1, 2, and 3, respectively.

Serial Data Buffer

The Serial Data Buffer is actually two separate registers, a transmit buffer and a receive buffer register. When data is moved to SBUF, it goes to the transmit buffer where it is held for serial transmission. (Moving a byte to SBUF is what initiates the transmission.) When data is moved from SBUF, it comes from the receive buffer.

Timer Registers

Register pairs (TH0, TL0), (TH1, TL1), and (TH2, TL2) are the 16-bit counting registers for Timer/Counters 0, 1, and 2, respectively.

Capture Registers

The register pair (RCAP2H, RCAP2L) are the capture registers for the Timer 2 "capture mode." In this mode, in response to a transition at the 8052's T2EX pin, TH2 and TL2 are copied into RCAP2H and RCAP2L. Timer 2 also has a 16-bit auto-reload mode, and RCAP2H and RCAP2L, hold the reload value for this mode. More about Timer 2's features on page 2-12.

Control Registers

Special Function Registers IP, IE, TMOD, TCON, T2CON, SCON, and PCON contain control and status bits for the interrupt system, the timer/counters, and the serial port. They are described in later sections.

OSCILLATOR AND CLOCK CIRCUIT

XTAL1 and XTAL2 are the input and output of a single-stage on-chip inverter, which can be configured with off-chip components as a Pierce oscillator, as shown in Figure 2-3. The on-chip circuitry, and selection of off-chip components to configure the oscillator are discussed on page 2-30.

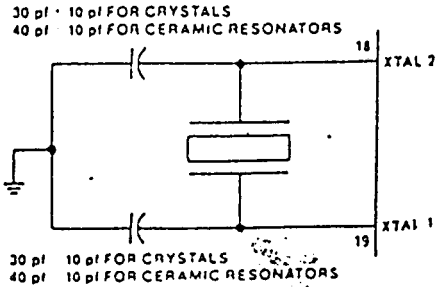


Figure 2-3. Crystal/Ceramic Resonator Oscillator

The oscillator, drives the internal clock generator, which provides the internal clocking signals to the chip. The internal clocking signals are at half the oscillator frequency, and define the internal phases, states, and machine cycles, described in the next section.

CPU TIMING

A machine cycle consists of six states (12 oscillator periods). Each state is divided into a Phase 1 half, during which the Phase 1 clock is active, and a Phase 2 half, during which the Phase 2 clock is active. Thus, a machine cycle consists of 12 oscillator periods, numbered S1P1 (State 1, Phase 1) through

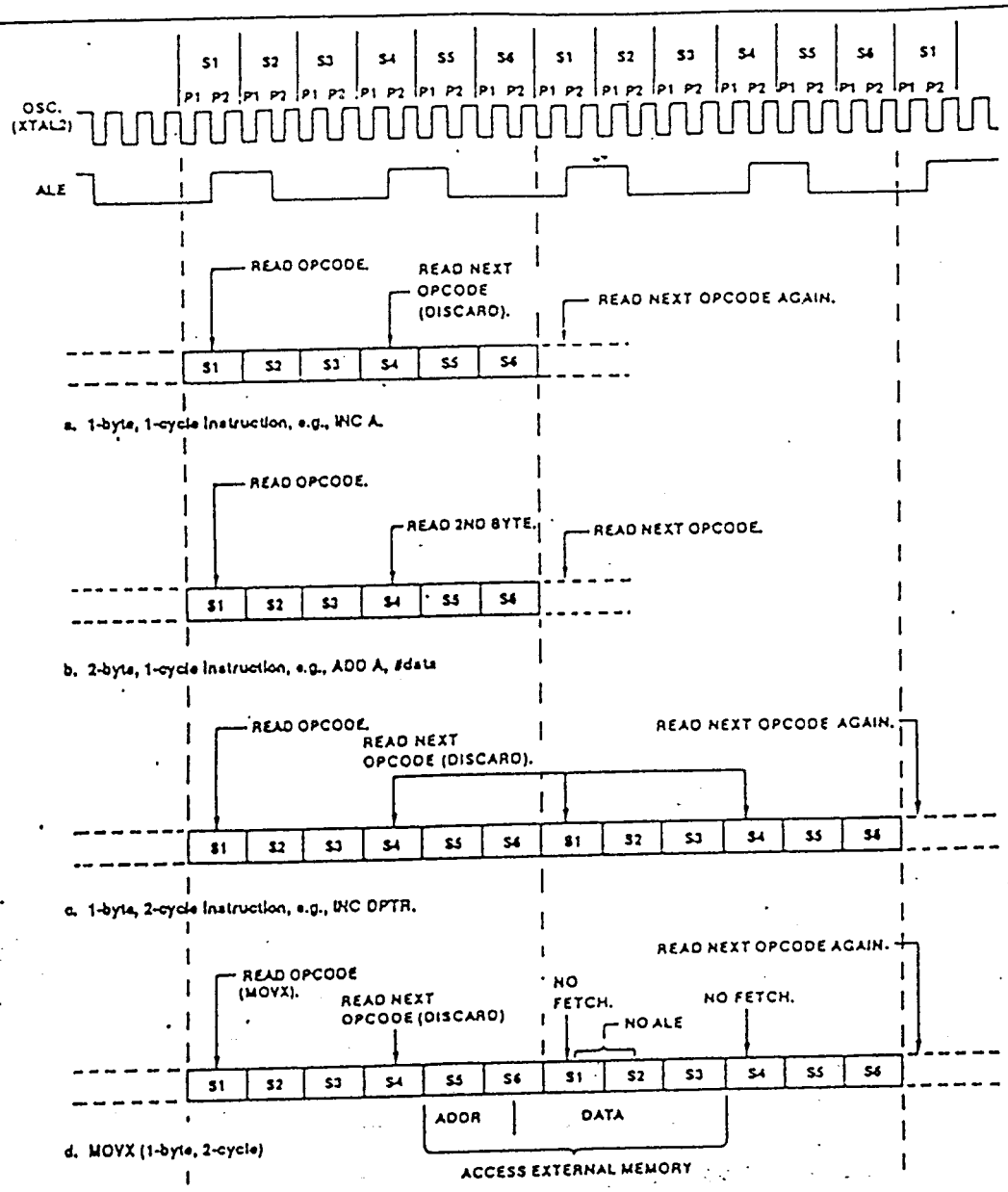


Figure 2-1. 8051 Fetch/Execute Sequences

S6P2 (State 6, Phase 2). Each phase last for one oscillator period. Each state lasts for two oscillator periods. Typically, arithmetic and logical operations take place during Phase 1 and internal register-to-register transfers take place during Phase 2.

The diagrams in Figure 2-4 show the fetch/execute timing referenced to the internal states and phases. Since these internal clock signals are not user accessible, the XTAL2 oscillator signal and the ALE (Address Latch Enable) signal are shown for external reference. ALE is normally activated twice during each machine cycle: one during S1P2 and S2P1, and again during S4P2 and S5P1.

Execution of a one-cycle instruction begins at S1P2, when the opcode is latched into the Instruction Register. If it is a 2-byte instruction, the second byte is read during S4 of the same machine cycle. If it is a 1-byte instruction, there is still a fetch at S4, but the byte read (which would be the next opcode) is ignored, and the Program Counter is not incremented. In any case, execution is complete at the end of S6P2. Figure 2-4a and 2-4b show the timing for a 1-byte, 1-cycle instruction and for a 2-byte, 1-cycle instruction.

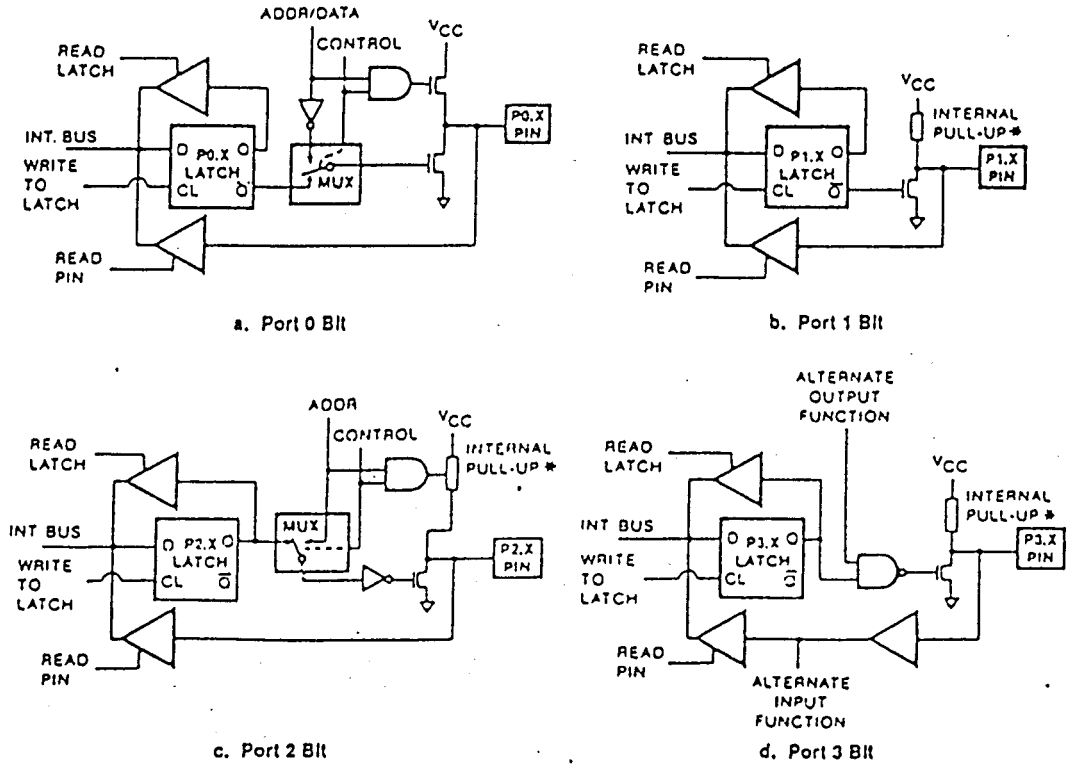
Most 8051 instructions execute in one cycle. MUL (multiply) and DIV (divide) are the only instructions that take more than two cycles to complete. They take four cycles.

Normally, two code bytes are fetched from Program Memory during every machine cycle. The only exception to this is when a MOVX instruction is executed. MOVX is a 1-byte 2-cycle instruction that accesses external Data Memory. During a MOVX, two fetches are skipped while the external Data Memory is being addressed and strobed. Figures 2-4c and 2-4d show the timing for a normal 1-byte, 2-cycle instruction and for a MOVX instruction.

PORT STRUCTURES AND OPERATION

All four ports in the 8051 are bidirectional. Each consists of a latch (Special Function Registers P0 through P3), an output driver, and an input buffer.

The output drivers of Ports 0 and 2, and the input buffers of Port 0, are used in accesses to external memory. In this application, Port 0 outputs the low byte of the external memory address, time-multiplexed with the byte being



*See Figure 2-6 for details of the internal pull up.

Figure 2-5. 8051 Port Bit Latches and I/O Buffers

written or read. Port 2 outputs the high byte of the external memory address when the address is 16 bits wide. Otherwise the Port 2 pins continue to emit the P2 SFR content.

All the Port 3 pins, and (in the 8052) two Port 1 pins are multifunctional. They are not only port pins, but also serve the functions of various special features as listed below:

PORT PIN	ALTERNATE FUNCTION
*P1.0	T2 (Timer/Counter 2 external input)
*P1.1	T2EX (Timer/Counter 2 capture/reload trigger)
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt)
P3.3	$\overline{INT1}$ (external interrupt)
P3.4	T0 (Timer/Counter 0 external input)
P3.5	T1 (Timer/Counter 1 external input)
P3.6	$\overline{WR}$ (external Data memory write strobe)
P3.7	$\overline{RD}$ (external Data memory read strobe)

*P1.0 and P1.1 serve these alternate functions only on the 8052.

The alternate functions can only be activated if the corresponding bit latch in the port SFR contains a 1. Otherwise the port pin is stuck at 0.

I/O Configurations

Figure 2-5 shows a functional diagram of a typical bit latch and I/O buffer in each of the four ports. The bit latch (one bit in the port's SFR) is represented as a Type D flip-flop, which will clock in a value from the internal bus in response to a "write to latch" signal from the CPU. The Q output of the flip-flop is placed on the internal bus in the response to a "read latch" signal from the CPU. The level of the port pin itself is placed on the internal bus in response to a "read pin" signal from the CPU. Some instructions that read a port activate the "read latch" signal, and others activate the "read pin" signal. More about that on page 2-8.

As show in Figure 2-5, the output drivers of Ports 0 and 2 are switchable to an internal ADDR and ADDR/DATA bus by an internal CONTROL signal for use in external memory accesses. During external memory accesses, the P2 SFR remains unchanged, but the P0 SFR gets 1s written to it.

Also show in Figure 2-5, is that if a P3 bit latch contains a 1, then the output level is controlled by the signal labeled "alternate output functions." The actual P3.X pin level is always available to the pin's alternate input function, if any.

Ports 1, 2, and 3 have internal pull-ups. Port 0 has open-drain outputs. Each I/O line can be independently used as an input or an output. (Ports 0 and 2 may not be used as general purpose I/O when being used as the ADDR/DATA BUS.) To be used as an input, the port bit latch must contain a 1, which turns off the output driver FET. Then, for Ports 1, 2, and 3, the pin is pulled high by the internal pull-up, but can be pulled low by an external source.

Port 0 differs in not having internal pullups. The pullup FET in the P0 output driver (see Figure 2-5a) is used only when the Port is emitting 1s during external memory accesses. Otherwise the pullup FET is off. Consequently P0 lines that are being used as output port lines are open drain. Writing a 1 to the bit latch leaves both output FETs off, so the pin floats. In that condition it can be used as a high-impedance input.

Because Ports 1, 2, and 3 have fixed internal pullups they are sometimes called "quasi-bidirectional" ports. When configured as inputs they pull high and will source current (I_{IL}, in the data sheets) when externally pulled low. Port 0, on the other hand, is considered "true" bidirectional, because when configured as an input it floats.

All the port latches in the 8051 have 1s written to them by the reset function. If a 0 is subsequently written to a port latch, it can be reconfigured as an input by writing a 1 to it.

Writing to a Port

In the execution of an instruction that changes the value in a port latch, the new value arrives at the latch during S6P2 of the final cycle of the instruction. However, port latches are in fact sampled by their output buffers only during Phase 1 of any clock period. (During Phase 2 the output buffer holds the value it saw during the previous Phase 1.) Consequently, the new value in the port latch won't actually appear at the output pin until the next Phase 1, which will be at S1P1 of the next machine cycle.

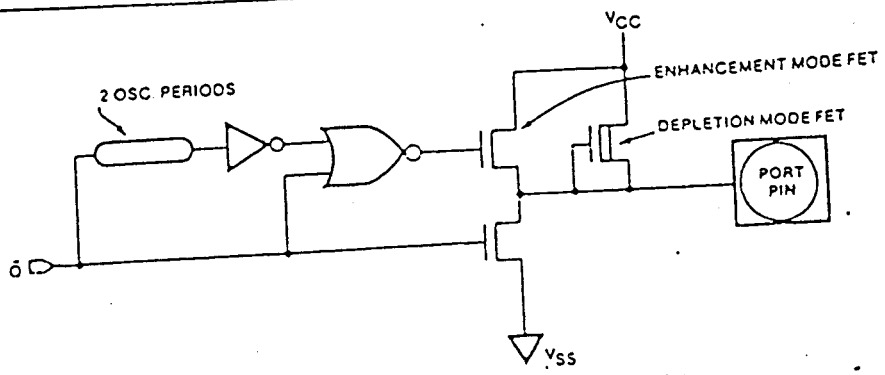
If the change requires a 0-to-1 transition in Port 1, 2, or 3, an additional pull-up is turned on during S1P1 and S1P2 of the cycle in which the transition occurs. This is done to increase the transition speed. The extra pull-up can source about 100 times the current that the normal pull-up can. It should be noted that the internal pull-ups are field-effect transistors, not linear resistors. The pull-up arrangements are shown in Figure 2-6.

In NMOS versions of the 8051, the fixed part of the pull-up is a depletion-mode transistor with the gate wired to the source. This transistor will allow the pin to source about 0.25 mA when shorted to ground. In parallel with the fixed pull-up is an enhancement-mode transistor, which is activated during S1 whenever the port bit does a 0-to-1 transition. During this interval, if the port pin is shorted to ground, this extra transistor will allow the pin to source an additional 30 mA.

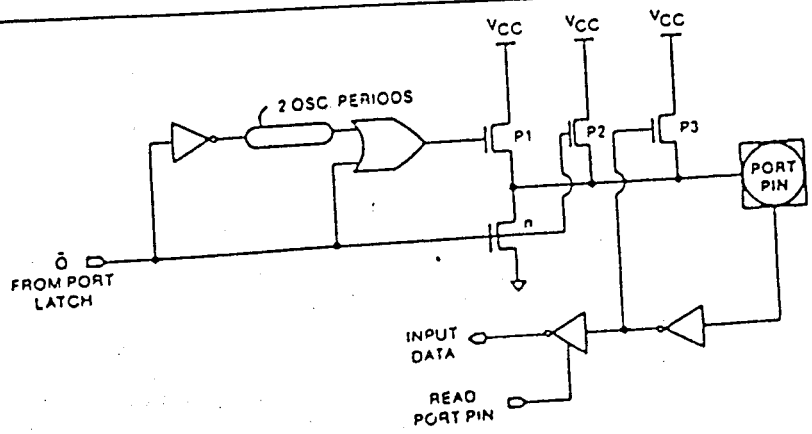
In the CMOS versions, the pull-up consists of three pFETs. It should be noted that an n-channel FET (nFET) is turned on when logical 1 is applied to its gate, and is turned off when a logical 0 is applied to its gate. A p-channel FET (pFET) is the opposite: it is on when its gate sees a 0, and off when its gate sees a 1.

Transistor pFET 1 in Figure 2-6 is turned on for two oscillator periods after a 0-to-1 transition in the port latch. While it's on, it turns on pFET 3 (a weak pull-up) through the inverter. This inverter and pFET 3 form a latch which holds the 1.

Note that if the pin is emitting a 1, a negative glitch on the pin from some external source can turn off pFET 3, causing the pin to go into a float state; pFET 2 is a very weak pull-up which is on whenever the nFET is off, in traditional CMOS style. It's only about 1/10 the strength of pFET 3. It's function is to restore a 1 to the pin in the event the pin *had* a 1 and lost it to a glitch.



a. NMOS Configuration



b. CMOS Configuration

Figure 2-6. Ports 1 and 3 NMOS and CMOS Internal Pull-up Configurations. (Port 2 is similar except that it holds the strong pull-up on while emitting 1s that are address bits.)

Port Loading and Interfacing

The output buffers of Ports 1, 2, and 3 can each drive four LS TTL inputs. These ports on NMOS versions can be driven in a normal manner by any TTL or NMOS circuit. Both NMOS and CMOS pins can be driven by open-collector and open-drain outputs, but note that 0-to-1 transitions will not be fast. In the NMOS device, if the pin is driven by an open collector output, a 0-to-1 transition will have to be driven by the relatively weak depletion mode FET in Figure 2-6a. In the CMOS device, an input 0 turns off pull-up pFET3, leaving only the very weak pull-up pFET2 to drive the transition.

Port 0 output buffers can each drive 8 LS TTL inputs. They do, however, require external pull-ups to drive NMOS inputs, except when being used as the ADDRESS/DATA bus.

Read-Modify-Write Feature

Some instructions that read a port, also read the latch, and others read the pin. Which ones do which? The instructions that read the latch rather than the pin are the ones that read a value, possibly change it, and then rewrite it to the latch. These are called "read-modify-write" instructions, listed below. When the destination operand is a port or a port bit, these instructions read the latch rather than the pin:

ANL	(logical AND, e.g., ANL P1,A)
ORL	(logical OR, e.g., ORL P2,A)
XRL	(logical EX-OR, e.g., XRL P3,A)
JBC	(jump if bit = 1 and clear bit, e.g., JBC P1.1, LABEL)
CPL	(complement bit, e.g., CPL P3.0)
INC	(increment, e.g., INC P2)
DEC	(decrement, e.g., DEC P2)
DJNZ	(decrement and jump if not zero, e.g., DJNZ P3, LABEL)
MOV PX.Y,C	(move carry bit to bit Y of Port X)
CLR PX.Y	(clear bit Y of Port X)
SET PX.Y	(set bit Y of Port X)

It is not obvious that the last three instructions in this list are read-modify-write instructions, but they are. They read the port byte, all 8 bits, modify the addressed bit, then write the new byte back to the latch.

The reason that read-modify-write instructions are directed to the latch rather than the pin is to avoid a possible misinterpretation of the voltage level at the pin. For example, a port bit might be used to drive the base of a transistor. When a 1 is written to the bit, the transistor is turned on. If the CPU then reads the same port bit at the pin rather that the latch, it will read the base voltage of the transistor and interpret it as a 0. Reading the latch rather than the pin will return the correct value of 1.

ACCESSING EXTERNAL MEMORY

Accesses to external memory are of two types: accesses to external Program Memory and accesses to external Data Memory. Accesses to external Program Memory use signal $\overline{PSEN}$ (program store enable) as the read strobe. Accesses to external Data Memory use $\overline{RD}$ or $\overline{WR}$ (alternate functions of P3.7 and P3.6) to strobe the memory.

Fetches from external Program Memory always use a 16-bit address. Accesses to external Data Memory can use either a 16-bit address (MOVX @DPTR) or an 8-bit address (MOVX @Ri).

Whenever a 16-bit address is used, the high byte of the address comes out on Port 2, where it is held for the duration of the read or write cycle. Note that the Port 2 drivers use the strong pullups during the entire time that they are emitting address bits that are 1s. This is during the execution of a MOVX @DPTR instruction. During this time the Port 2 latch (the Special Function Register) does not have to contain 1s, and the contents of the Port 2 SFR are not modified. If the external memory cycle is not immediately followed by another external memory cycle, the undisturbed contents of the Port 2 SFR will reappear in the next cycle.

If an 8-bit address is being used (MOVX @Ri), the contents of the Port 2 SFR remain at the Port 2 pins throughout the external memory cycle. This will facilitate paging.

In any case, the low byte of the address is time-multiplexed with the data byte on Port 0. The ADDR/DATA signal drives both FETs in the Port 0 output buffers. Thus, in this application the Port 0 pins are not open-drain outputs, and do not require external pull-ups. Signal ALE (address latch enable) should be used to capture the address byte into an external latch. The address byte is valid at the negative transition of ALE. Then, in a write cycle, the data byte to be written appears on Port 0 just before $\overline{WR}$ is activated, and remains there until after $\overline{WR}$ is deactivated. In a read cycle, the incoming byte is accepted at Port 0 just before the read strobe is deactivated.

During any access to external memory, the CPU writes 0FFH to the Port 0 latch (the Special Function Register), thus obliterating whatever information the Port 0 SFR may have been holding.

The TCM3105

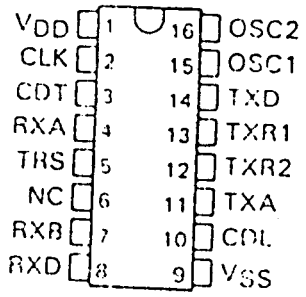
The TCM3105 provides a majority of the functions required of a medium speed FSK modem in a single 16-pin DIP. The device is manufactured using silicon-gate complementary MOS technology. The TCM3105 features single 5 V supply operation and typical power consumption of approximately 40 mW. This makes the device ideally suited for use in battery operated equipment applications, as well as in standard applications. The TCM3105 device pinout is shown in Figure 2. Refer to pin description listed in Table 1 for the function and significance of each pin.

The TCM3105 is characterized for operation from 0 °C to 70 °C (JL suffix) as well as over the extended free-air temperature range of −40 °C to 85 °C (JE suffix).

Table 1. TCM3105 Pinout Description

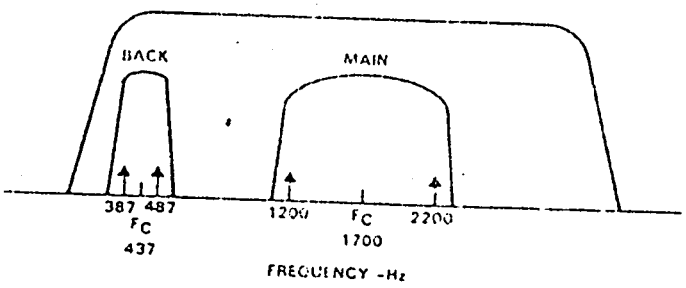
PIN NAME	PIN NO.	I/O	DESCRIPTION
CDL	10	I	Carrier Detect Level—Sets the threshold level for the carrier detect decision. Refer to Description of the Carrier Detect Adjustment paragraph.
CDT	3	O	Carrier Detect — A high logic level output indicates the presence of a carrier at the RXA pin.
CLK	2	O	Clock — Continuous output clock signal at 16 times the highest selected transmit or receive baud rate.
OSC1 OSC2	15 16		Oscillator 1 and 2 — Input connections for external 4.4336 MHz crystal. See Table 2 for list of crystal manufacturers. If an external clock input is provided, then the OSC1 pin is left open and the clock is connected to the OSC2 pin.
RXA	4	I	Receive Analog — This input is referenced to an internal voltage and must be ac coupled.
RXB	7	I	Receive Bias Adjust — This input sets the threshold level of the slicer that allows the bias distortion on the RXD pin to be minimized. Refer to Description of the Receive Bias Adjustment paragraph.
RXD	8	O	Receive Digital Output — Outputs the demodulated receive data in positive logic, i.e., a mark is indicated by a high level and a space is indicated by a low level. The RXD output pin will remain high if there is no analog input on the RXA pin.
TRS	5	I	Transmit/Receive Standard Select Input — This pin along with TXR1 and TXR2 select the standard and mode to be used. See Table 1.
TXD	14	I	Transmit Digital — Digital input to the modulator in positive logic, i.e., a mark is indicated by a high level and a space is indicated by a low level. The data can be accepted at any rate from zero up to the selected baud rate and may be totally asynchronous.
TXR1 TXR2	13 12	I I	Transmit Rate 1 and 2 — These signals along with TRS set the standard and mode to be used. See Table 1.
V_{DD}	1		Positive supply voltage — 5 volts nominal

J DUAL-IN-LINE PACKAGE
(TOP VIEW)

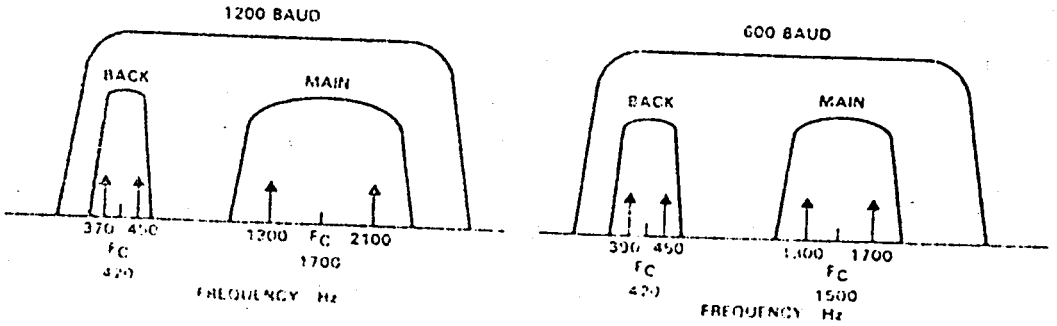


NC - No internal connection

Figure 2. TCM3105 Pinout



(a) Bell 202 Channel Assignments



(b) CCITT V.23 Channel Assignments

Figure 3. Bell 202 & CCITT V.23 Channel Assignments

Modes of Operation of the TCM3105

The TCM3105 is an FSK type modem that is designed to implement the Bell 202 and CCITT V.23 Standards (see Figure 3) which define mark and space frequencies, and the maximum data rate that can be transmitted for a given mark/space pair.

The TCM3105 can be placed into a given mode of operation by applying the proper signals to the TRS (Transmit/Receive Select) and the TXR1 and TXR2 (Transmit Rate 1 and 2) input pins. The various modes of operation for the modem are summarized in Table 2. The CLK signal pin operates at a clock frequency of 16 times that of the selected receive or transmit bit rate, whichever is higher.

Table 2. TCM3105 Modes of Operation

Standard	TRS	TXR1	TXR2	Transmit Bit Rate (Bit/s)	Receive Bit Rate (Bit/s)	Transmit Freq (Hz)	Receive Freq (Hz)	Clock (MHz)
CCITT V.23	0	0	0	1200	1200	M 1400 S 2100	M 1300 S 2100	19.11
	1	0	0	1200	75	M 1300 S 2100	M 390 S 450	19.11
	0	0	1	600	75	M 1380 S 1740	M 390 S 450	9.56
	1	0	1	600	600	M 1380 S 1740	M 1380 S 1740	9.56
	0	1	0	75	1200	M 390 S 450	M 1300 S 2100	19.11
	1	1	0	75	600	M 390 S 450	M 1300 S 1700	9.56
	0	1	1	75	75	M 390 S 450	M 390 S 450	1.19
BELL 202	CLK	0	0	1200	1200	M 1200 S 2200	M 1200 S 2200	19.11
	CLK/8	0	1	1200	150	M 1200 S 2200	M 387 S 437	19.11
	CLK/3	0	1	1200	5	M 1200 S 2200	M 387 S 0	19.11
	CLK	1	0	150	1200	M 387 S 437	M 1200 S 2200	19.11
	CLK	1	1	150	150	M 387 S 487	M 387 S 487	2.39
	*	1	*	5	1200	M 387 S 0	M 1200 S 2200	19.11
	1	1	1	Transmit Disabled	1200	Transmit Disabled	M 1200 S 2200	19.11

*In this mode, the modulation is controlled by the TXR1 and TXR2 inputs. TXD is set to 1

If TRS = CLK & TXR2 = 0, then TXA = 387 Hz
If TRS = 1 & TXR2 = 1, then TXA = 0 Hz

Architectural Description of the TCM3105

The modem has four main functional blocks: a transmitter, a receiver, a [illegible] detector, and timing and control (see Figure 4).

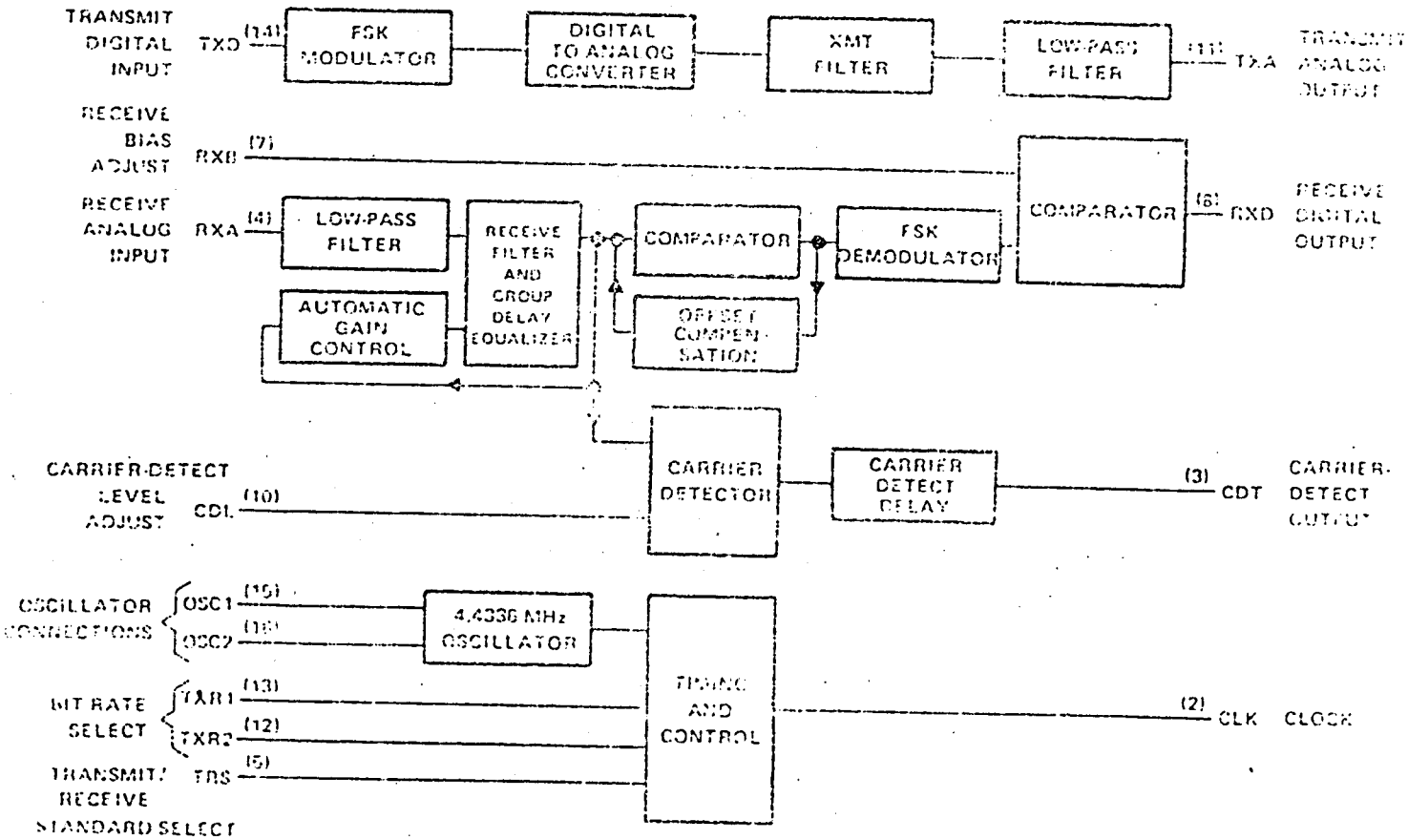
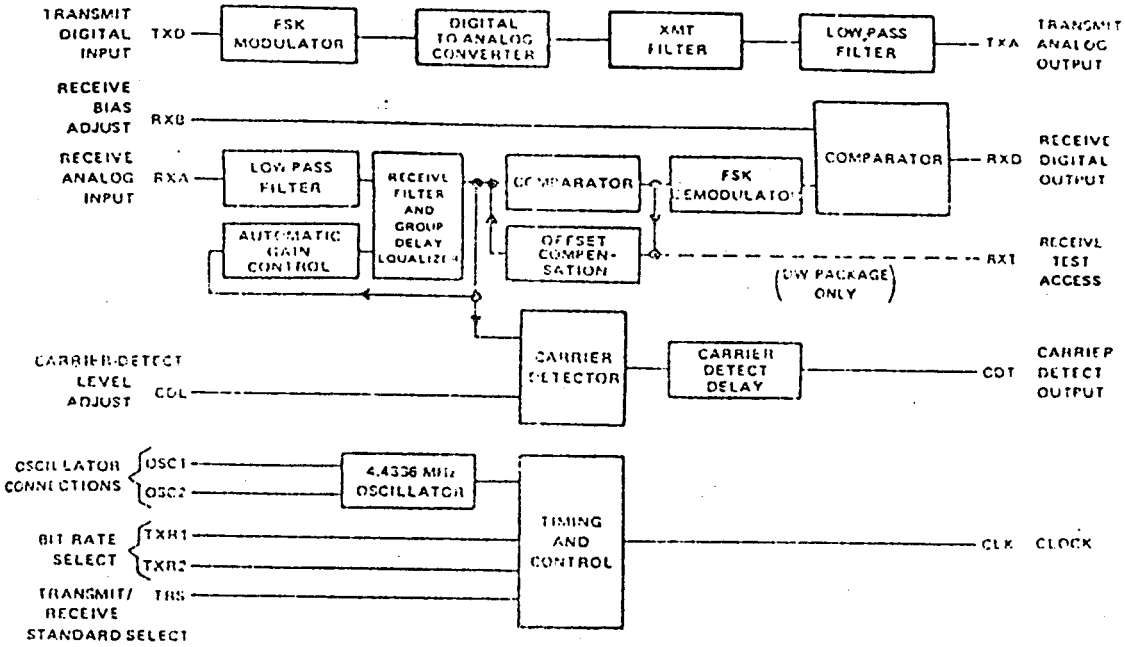
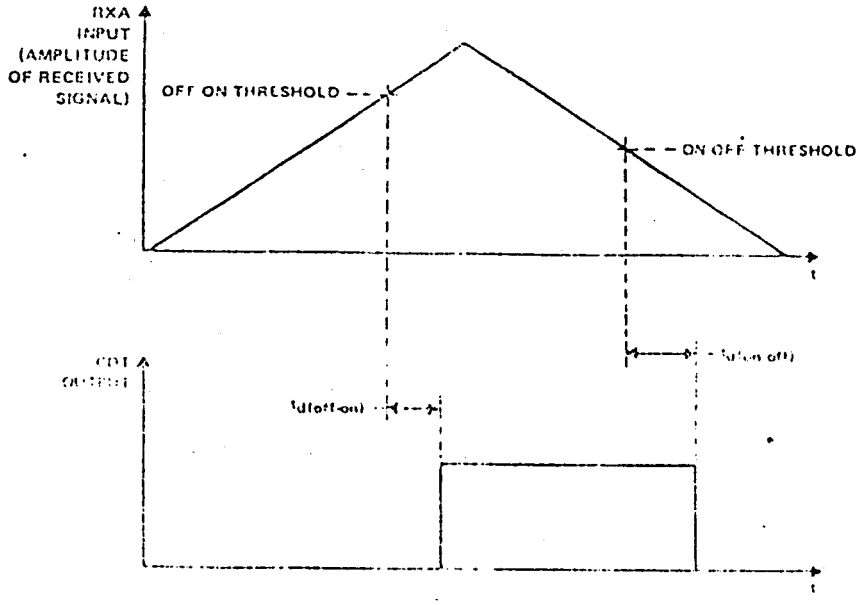


Figure 4. TCM3105 Functional Block Diagram

functional block diagram



timing diagram



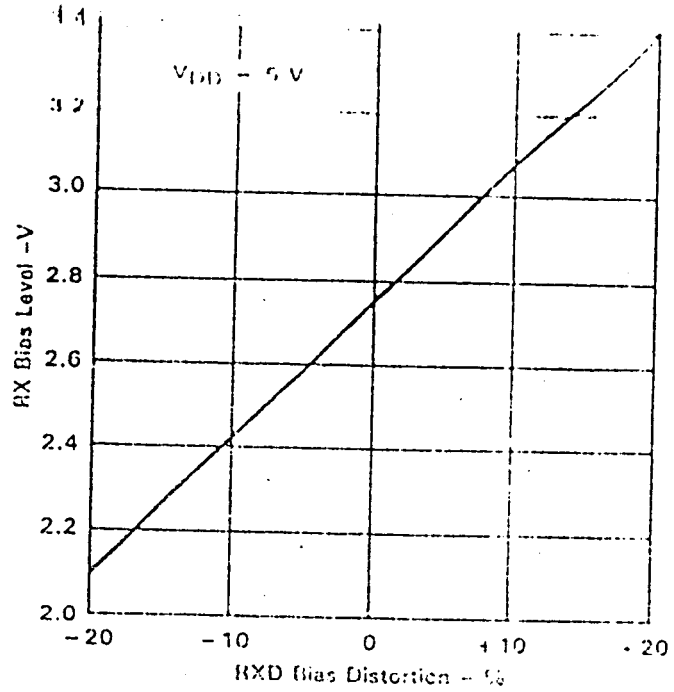


Figure 8. RXB Bias Level vs RXD Bias Distortion

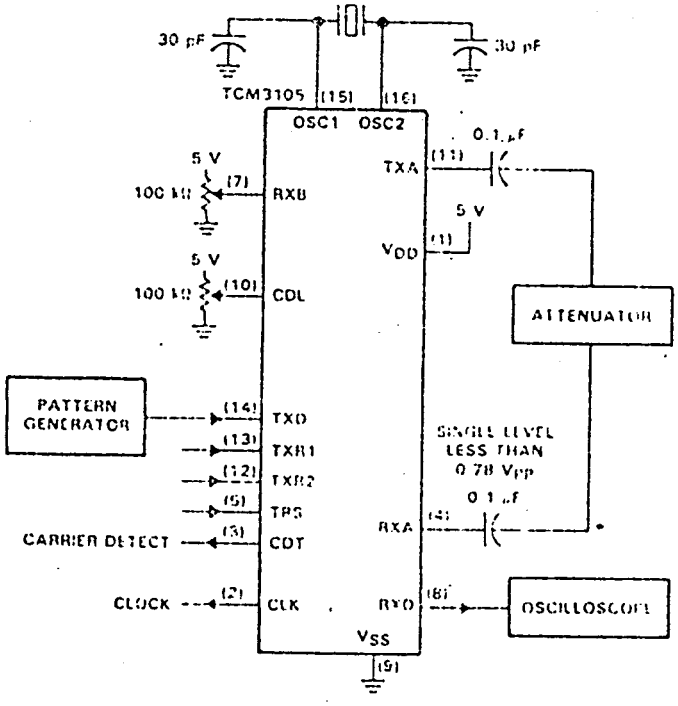


Figure 9. Loopback Circuit Diagram

Basic Principles of Modem Operation

A modem is a device that enables two digital electronic systems to communicate over the telephone network. To accomplish this, the digital signals must be converted to analog signals. The short pulses used by digital equipment contain high frequency components that are not supported by the limited bandwidth of telephone networks (300 Hz - 3400 Hz).

There are two major schemes of modulation used in modems for telephone networks. These are Frequency Shift Keying (FSK) and Phase Shift Keying (PSK). In FSK, serial data is modulated so that a "mark" is represented by a sine wave of one frequency, and a "space" is represented by a different frequency. In PSK, transitions in the digital bit stream are represented as shifts in phase angle of a single carrier frequency. The FSK concept is used by the TCM3105. The telephone network is a single-twisted pair of wires, usually 24 or 26 gauge. Two separate paths of communication are required by digital equipment systems in order to communicate with each other. Each system must have transmit and receive capability. This interface is supplied by the modem. Full duplex operation is the simultaneous transmission and reception on a single pair of wires. A two-to-four-wire converter, or hybrid as it is called in the telecommunications industry, is required (see Figure 1). This device removes the transmitted data from the receive path so that transmitted data does not interfere with valid received data.

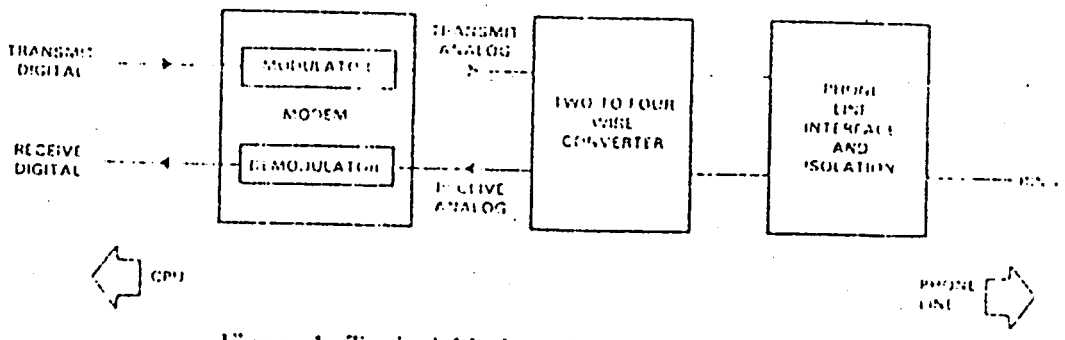


Figure 1. Typical Modem System Configuration

The two-to-four wire converter requires matching the ranges of telephone network impedances. The impedance of the telephone network varies from line to line due to manufacturing and installation tolerances of the communications hardware. It is difficult to obtain good cancellation in a mass produced piece of equipment.

Four separate frequencies, two transmit and two receive, are used to properly balance the two- to four-wire converter. This is to ensure that transmitted and received data do not interfere with each other.

INFORMATION TO USERS

WARNING: THIS EQUIPMENT HAS BEEN CERTIFIED TO COMPLY WITH THE LIMITS FOR A CLASS B COMPUTING DEVICE, PURSUANT TO SUBPART J OF PART 15 OF FCC RULES. ONLY COMPUTER AND PERIPHERALS (COMPUTER INPUT/ OUTPUT DEVICES, TERMINALS, PRINTERS, ETC.) CERTIFIED TO COMPLY WITH THE CLASS B LIMITS MAY BE ATTACHED TO THIS KEYBOARD. OPERATION WITH NON-CERTIFIED PERIPHERALS IS LIKELY TO RESULT IN INTERFERENCE TO RADIO AND TV RECEPTION.

To insure compliance to FCC non-interference regulations, peripherals attached to this computer require shielded I/O cables.

This equipment generates and uses radio frequency energy and if not installed and used proporly, that is, in strick accordance with the manufacturer's instructions, may cause interference to radio and television reception. It has been type tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

1. Reorient the receiving antenna
2. Relocate the computer with respect to the receiver
3. Move the computer away from the receiver
4. Plug the computer into a different outlet so that computer and receiver are on different branch circuits.

If necessary, the user should consult the dealer or an experienced radio/television technician for additional suggestions. The user may find the following booklet prepared by the FCC helpful:

"How to Indentify and Resolve Radio-TV Interference Problems"

This Booklet is available from the U.S. Government Printing Office, Washington, D.C. 20402- Stock No. 004-000-00345-4.

NOTICE: In order to insure continued compliance to the FCC emission limits for this keyboard, it is necessary to use computer and I/O cables which are shielded. The shield must be terminated to the metallic cabinet at both ends to guarantee adequate suppression of undesireable emissions.

INTRODUCTION

The BTC-5060 keyboard is a direct replacement for the IBM*PC, XT and AT Personal Computer keyboards.

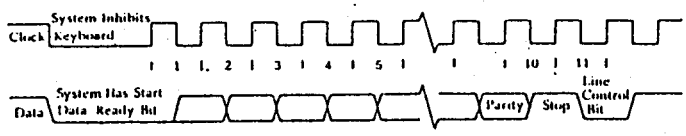
There are no software modifications or special interfaces need. Using it is the same as described in the IBM* Personal Computer Guide to Operations-Handbook you received with your Personal Computer.

ELECTRICAL DATA (for AT)

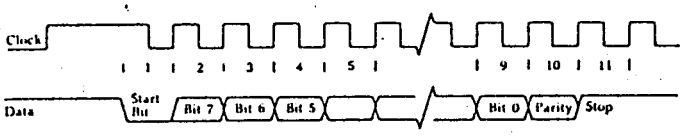
INPUT/OUTPUT

The keyboard will check the status of the "Clock" line and the "Data" line prior to every data transmission. If the "Clock" line is low, then keyboard is disabled; If the "Clock" line is high but the "Data" line is low, then do keyboard input from the PC/AT. If both line is high, then the keyboard will initiate transmission by set the "Data" line low and clocking the start bit to the host computer on the falling edge of the clock pulse. The keyboard will then clock the 8 data bits to the host computer on the falling edge of each clock pulse. While each clocking the keyboard will check if the "Clock" line is still in high, otherwise the keyboard will give up the data transmition. If the transmition is OK, then the keyboard will output the odd parity bit and raise the "Data" line to high then clocking the stop bit to complete the transmission.

DATA GOING INTO KEYBOARD



DATA COMING FROM KEYBOARD



* IBM is a trademark of International Business Machines CORP.

KEY OPERATION

All keys on the keyboard are Make/Break and repeatable. The most significant bit of each scan code is a low level key for key depressions and a high level for key releases.

When a key is depressed, the keyboard will transmit its assigned scan code. If the key is held down for a repeat delay time, its scan code will be transmitted at a assigned repeat rate (the repeat delay and rate is variable) for as long as the key is held depressed or until another key is operated.

Depression of a second key will cause the first key to stop repeating and the scan code of the second key to be transmitted, as well as initiation of a delay sequence.

LED OPERATION

First depression of the key turns ON the LED. The second depression turns OFF the LED and so on toggle the LED's. LED's are OFF on power-up and software Reset.

KEYSTROKE BUFFER

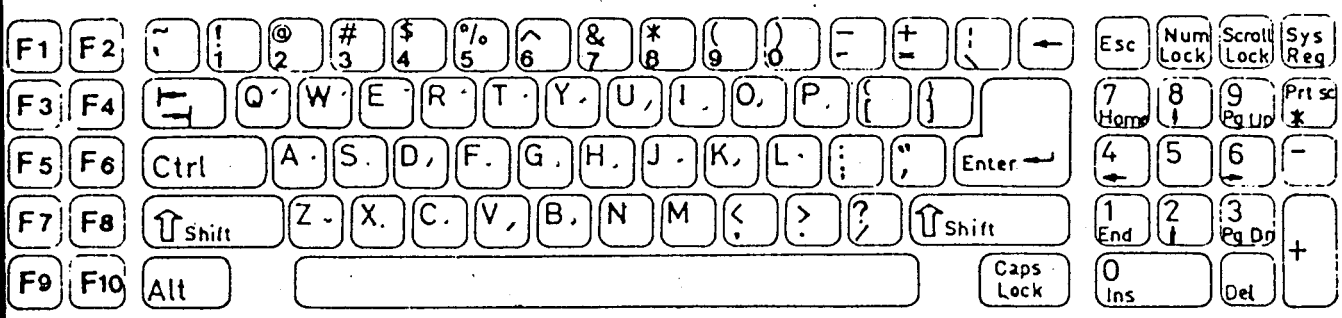
A 16-character (First-in, First-out) keystroke buffer is provided to prevent loss of keystrokes. An "00 HEX" will be inserted into the overflow buffer if the keystroke overflows. the keyboard will transmit this code once it has reached the top of the buffer.

KEYBOARD DIAGNOSTIC

The keyboard microprocessor will perform a diagnostic self-test after Power-Up or after the host system signals the keyboard to perform a software Reset by "Reset" command from the host system. The microprocessor will check its data memory locations, do a sum-check, internal ram check and check for any depressed keys. If the diagnostic test is correct, the keyboard will transmit an "AA HEX" code. This will be the first transmission following a Power-Up condition. If the diagnostic test was unsuccessful, then the keyboard will transmit an "FD/FC HEX" code. In either case, after the keyboard diagnostic check the keyboard will begin normal operation.

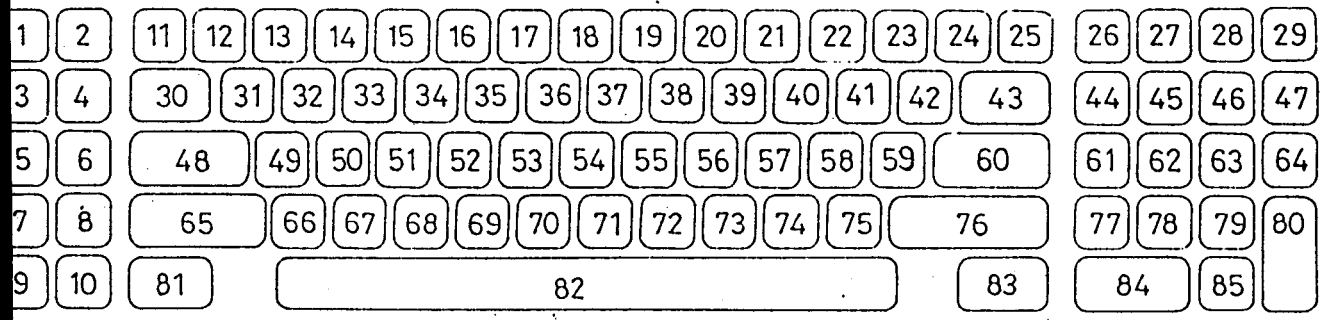
CHARACTER ARRAY ASSIGNMENTS

The layout of this keyboard is completely same as that of IBM* PC/AT keyboard, but PC and PC/XT can correspond to it also.



KEY POSITIONS

Please refer to upper left of CIRCUIT DIAGRAM shown on page 7



KEYSWITCH SCAN CODE

The keys shown in the code chart below will output a 8-bit code for a depression (make) or a release (break) and are electrical compatible for the IBM* personal computer PC and XT (shown on fig 1), AT (shown on fig 2).

The bit assignments for each key are shown in the hexadecimal numbering system. Hexadecimal means 16 and is simply a shorthand notation used to express the binary bit patterns.

As to AT, the break code of each key is transmitted a "F0 HEX" followed by that key's make code. For example, the make code of character "A" is "1C HEX", so its break code is "F0, 1C HEX".

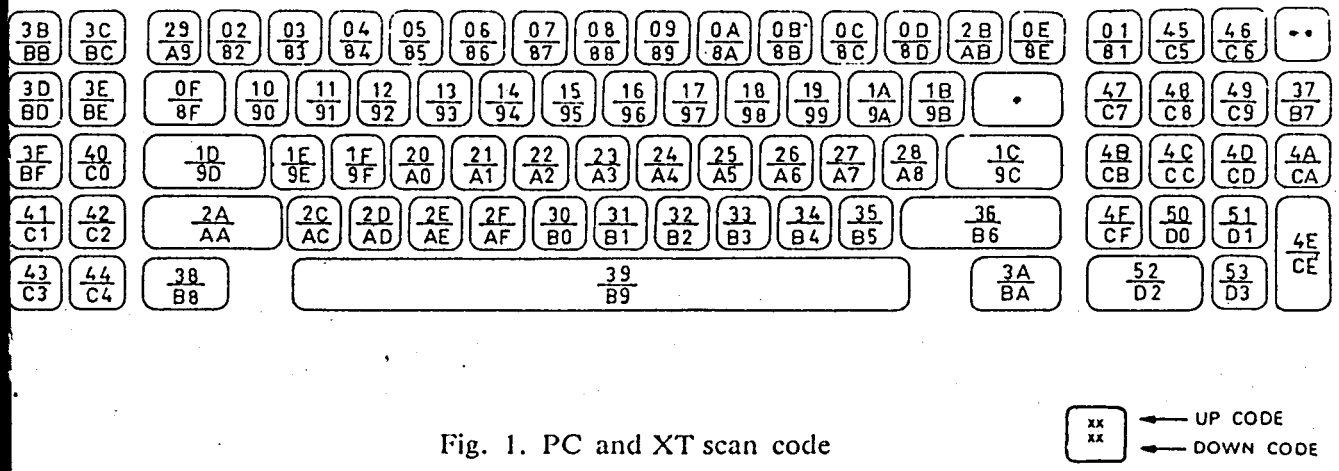


Fig. 1. PC and XT scan code

Notes:

• = No code generate.

•• = When this key is pressed with "Ctrl" simultaneously, It will generate function equal to "Ctrl" + "Alt" + "Del".

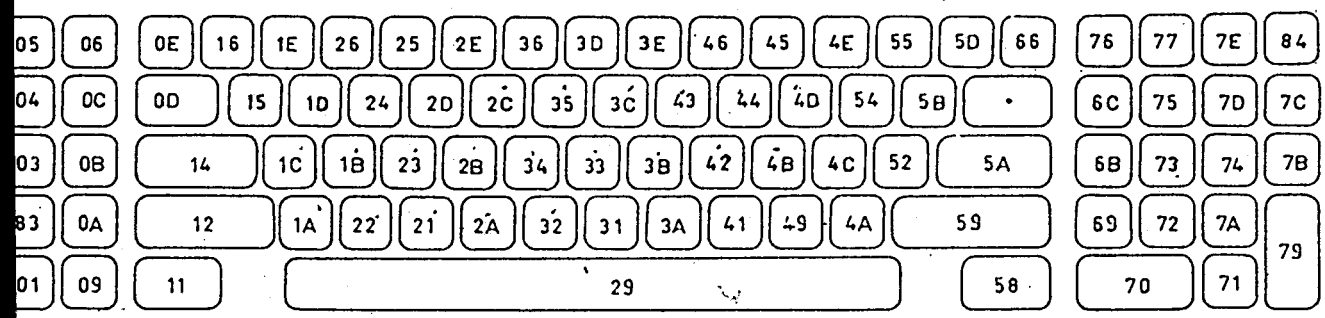
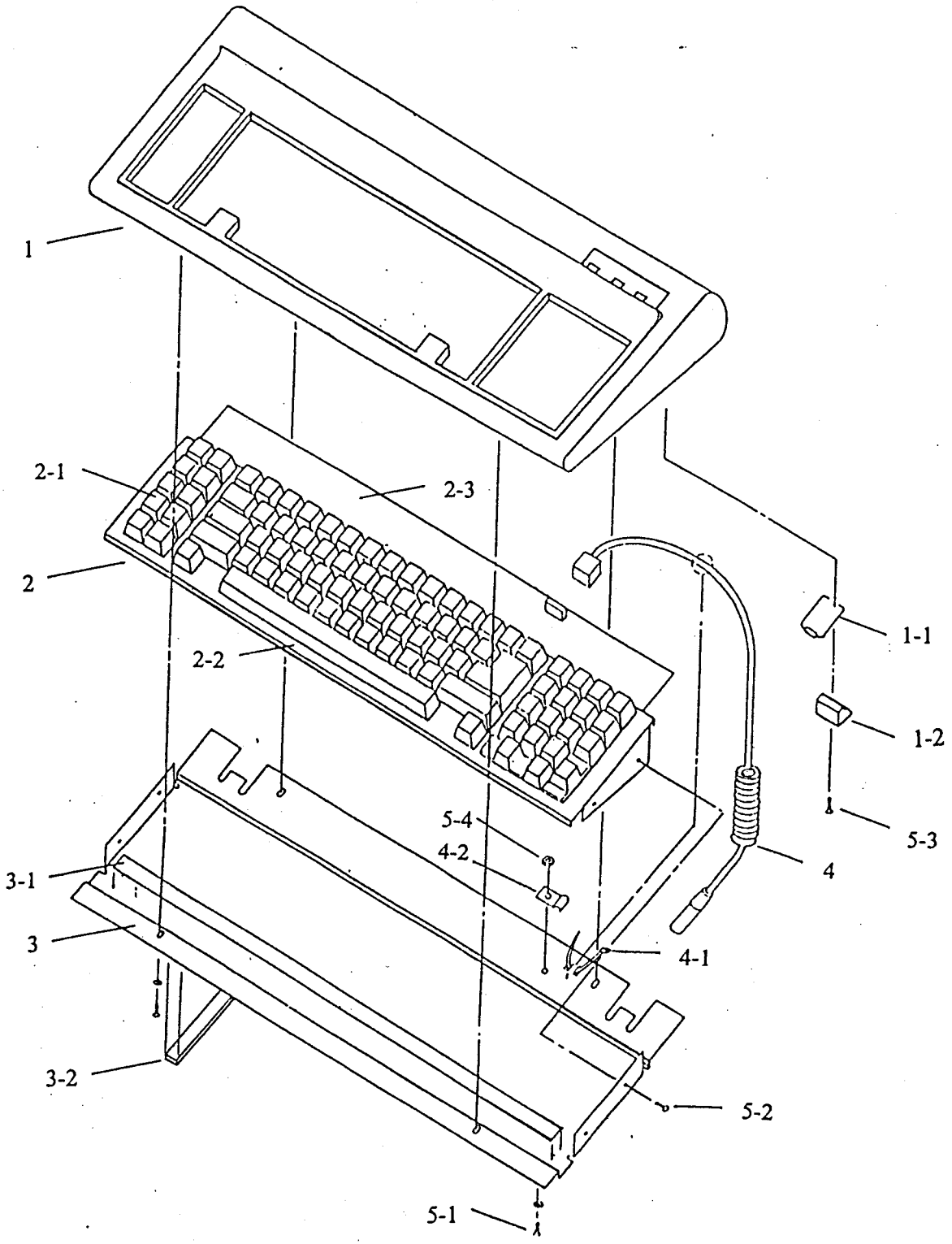


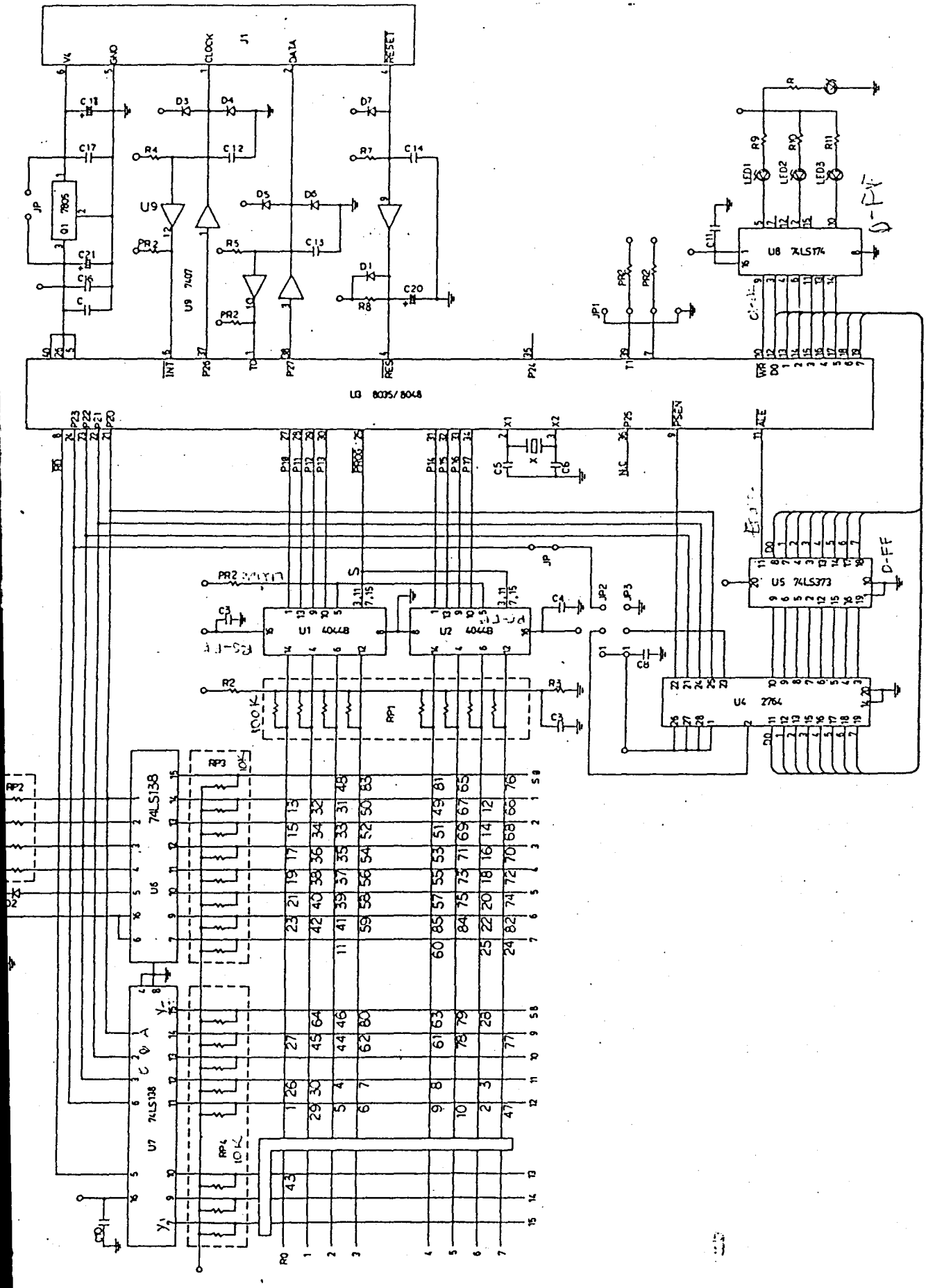
Fig 2. AT scan code

BTC-5060 KEYBOARD ASSEMBLY



1 Enclosure
1-1 Leg, Adjustable
1-2 Eclosure, Bail Block
2 Keyboard assembly
2-1 Keytop Set
2-2 Mounting Plate
2-3 Printed Circuit Board
3 Base Plate
3-1 Strip, Adhesive
3-2 Foot, Rubber Black
4 Cable Assembly
4-1 Cable Tie No 4 inches
4-2 Cable Clip
5 Mounting Hardward
5-1 Screw M4 x 6 p = 0.7
5-2 Screw 11/ 8-40 x 0.312
5-3 Screw M3 x 12 p = 0.5
5-4 NUT M4. P = 0.7

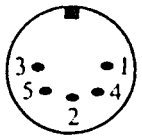
CIRCUIT DIAGRAM



PART LIST

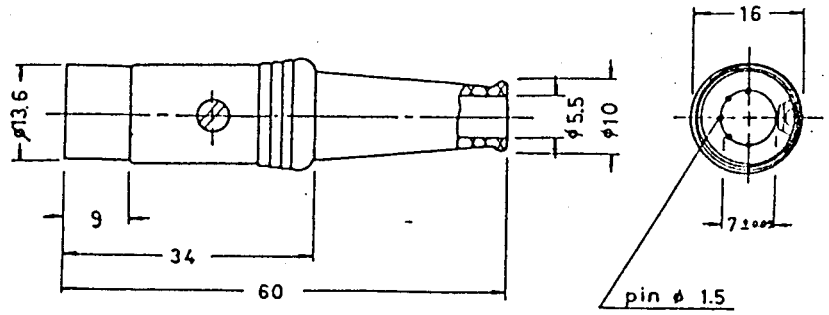
PARTS NO.	DESCRIPTION	PARTS NO.	DESCRIPTION
U1, U2	4044B	,R10,R11,R9	100 Ω ±5% ¼W
U3	C23282E/C34432E	RP1	100K Ω ±5% * 8
U6, U7	74LS138	RP2, RP3, RP4	10K Ω ±5% * 8
U8	74LS174	C1, C3, C4, C7	
U9	7407	C8,C10,C15	
D1,D2	1N4148	C17,C11	0.1μF 50V
LED1,LED2		C5, C6	10PF 50V
LED3	3mm RED	C12,C13,C14	33PF 50V
R2	Depends on 4044B	C18,C20	10μF 16V
R3	Depends on 4044B	X'TAL	6.144 MHz X'TAL
R4,R5,R7	3.3K Ω ±5% ¼W	J1	6 PIN Right Angle Wafer
R8	100K Ω ±5% ¼W		

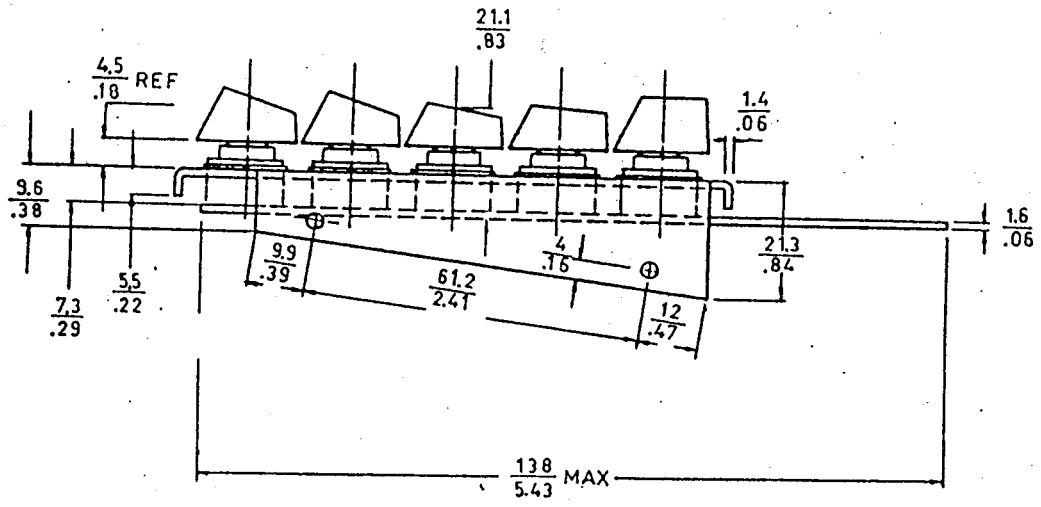
KEYBOARD INTERFACE CONNECTOR

DESCRIPTION	VOLTAGE	PINS	CONNECTOR
Keyboard Clock	+5VDC Signal	1	
Keyboard Data	+5VDC Signal	2	
Keyboard RESET	0	3	
Ground	0	4	
Power Supply	+5 VDC	5	
			5 Pin DIN at System Unit

MOUNTING DIMENSIONS

Cable assembly 5 pins male audio din plug, 180 degrees arrangement, each pin at 45 degrees, 5 conduct plus shield (# 24 AWG).





JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER

EL -1799 TUGAS AKHIR - 6 SKS

Nama Mahasiswa	:	**Nurlambang Atmojo**
Nomor Pokok	:	2912201782
Bidang Studi	:	Komputer
Tugas diberikan	:	22-3-1995
Tugas diselesaikan	:	
Dosen Pembimbing I	:	**Ir. Yoyon Kusnendar Suprapto, Msc.**
Dosen Pembimbing II	:	**Ir. Hanny Budinugroho**

JUDUL TUGAS AKHIR :

PERANCANGAN DAN PEMBUATAN INTERFACE PENGONTROL PINTU OTOMATIS UNTUK KARYAWAN DENGAN TEKNIK FREQUENCY SHIFT KEYING

URAIAN TUGAS AKHIR :

Di dalam sebuah pabrik, misalnya pabrik kimia, biasanya terdapat Laboratorium atau ruangan khusus. Dimana hanya orang-orang tertentu yang diijinkan masuk ke Laboratorium tersebut. Ini disebabkan dalam Laboratorium itulah awal mula perancangan dan pembuatan obat atau bahan kimia yang diproduksi oleh pabrik tersebut. Formula bahan kimia ini biasanya sangatlah dirahasiakan, terutama terhadap pabrik obat lainnya agar tidak dapat ditiru. Untuk itulah dibuat beberapa pengamanan dalam pabrik. Diantaranya pemasangan pintu khusus dengan "Kunci" Password atau ID Card atau pendeteksi yang lain. Bahkan seringkali digunakan gabungan dari dua atau lebih alat pendeteksi, misalnya ID Card dan kamera video.

Permasalahan itu masih mudah untuk diselesaikan bila dalam perusahaan itu hanya terdapat 1 buah Laboratorium atau ruangan khusus dengan satu pintu masuk. Tapi seringkali pabrik-pabrik kimia mempunyai ruangan-ruangan khusus yang lebih banyak, dan dengan pintu masuk yang untuk per-ruangannya juga lebih dari 1. Permasalahan yang lain adalah bagaimana bila ternyata pusat pengontrolan terhadap pintu-pintu tersebut letaknya sangat jauh.

Karena itu diperlukan suatu alat pengendali pintu-pintu ke ruangan di dalam pabrik, yang akan menentukan karyawan mana saja yang diperbolehkan untuk melewati pintu tersebut. Dimana agar berbeda, tiap karyawan dibedakan kodenya. Kode itu bisa berupa ID Card, atau Password yang diinputkan ke Keyboard.

Tugas akhir ini adalah merancang dan membuat pengontrol pintu otomatis yang selain mempunyai kemampuan menentukan boleh tidaknya karyawan itu melewati sebuah pintu otomat. Tiap-tiap karyawan yang melewati satu pintu otomat akan dicatat. Termasuk juga akan dicatat waktu masuk dan keluar tiap karyawan tersebut. Penentuan ini bergantung pula pada apakah karyawan tersebut telah di- PHK atau pindah bagian ke bagian lain.

Selain itu perihal yang juga harus diperhatikan adalah sangat tidak mungkin ada seorang karyawan memasuki 2 Laboratorium pada saat yang sama. Hal ini juga menjaga terhadap kemungkinan penyalahgunaan kunci password karyawan. Kemudian pula pengontrolan akan bisa jarak jauh (beberapa kilometer) untuk mengantisipasi jauhnya letak tiap-tiap pintu dan pusat pengontrolnya.

Dengan pembuatan Interface ini diharapkan dapat meningkatkan efisiensi perusahaan dalam mengontrol ketepatan dan keamanan ruangan-ruangan.

Surabaya, 22 Maret 1995

Menyetujui
Dosen Pembimbing I

Ir. Yoyon K. Suprapto, MSc.
NIP. 130 687 439

Dosen Pembimbing II

Ir. Hanny Budinugroho
NIP. 131 651 433

Mengetahui
Bidang Studi Teknik Komputer
Koordinator,

Ir. Yoyon K. Suprapto, MSc.
NIP. 130 687 439

Jurusan Teknik Elektro FTI - ITS
Ketua,

DR Ir. M. Salehudin, M. Eng.Sc.
NIP.130 532 027

USULAN TUGAS AKHIR

1. **Judul :** **PERANCANGAN DAN PEMBUATAN INTERFACE PENGONTROL PINTU OTOMATIS UNTUK KARYAWAN DENGAN TEKNIK FREQUENCY SHIFT KEYING**

2. **Ruang Lingkup :**
   - **Teknik Interfacing**
   - **Basis Data**
   - **Sistem Digital dan Mikroprosesor**

3. **Latar Belakang :** Di dalam sebuah pabrik, misalnya pabrik kimia, biasanya terdapat Laboratorium atau ruangan khusus. Dimana hanya orang-orang tertentu yang diijinkan masuk ke Laboratorium tersebut. Ini disebabkan dalam Laboratorium itulah awal mula perancangan dan pembuatan obat atau bahan kimia yang diproduksi oleh pabrik tersebut. Formula bahan kimia ini biasanya sangatlah dirahasiakan, terutama terhadap pabrik obat lainnya agar tidak dapat ditiru. Untuk itulah dibuat beberapa pengamanan dalam pabrik. Diantaranya pemasangan pintu khusus

dengan "Kunci" Password atau ID Card atau pendeteksi yang lain. Bahkan seringkali digunakan gabungan dari dua atau lebih alat pendeteksi, misalnya ID Card dan kamera video.

Permasalahan itu masih mudah untuk diselesaikan bila dalam perusahaan itu hanya terdapat 1 buah Laboratorium atau ruangan khusus dengan satu pintu masuk. Tapi seringkali pabrik-pabrik kimia mempunyai ruangan-ruangan khusus yang lebih banyak, dan dengan pintu masuk yang untuk per-ruangannya juga lebih dari 1. Permasalahan yang lain adalah bagaimana bila ternyata pusat pengontrolan terhadap pintu-pintu tersebut letaknya sangat jauh.

Karena itu diperlukan suatu alat pengendali pintu-pintu ke ruangan di dalam pabrik, yang akan menentukan karyawan mana saja yang diperbolehkan untuk melewati pintu tersebut. Dimana agar berbeda, tiap karyawan dibedakan kodenya. Kode itu bisa berupa ID Card,

atau Password yang diinputkan ke Keyboard.

Tugas akhir ini adalah merancang dan membuat pengontrol pintu otomatis yang selain mempunyai kemampuan menentukan boleh tidaknya karyawan itu melewati sebuah pintu otomat. Tiap-tiap karyawan yang melewati satu pintu otomat akan dicatat. Termasuk juga akan dicatat waktu masuk dan keluar tiap karyawan tersebut. Penentuan ini bergantung pula pada apakah karyawan tersebut telah di-PHK atau pindah bagian ke bagian lain.

Selain itu perihal yang juga harus diperhatikan adalah sangat tidak mungkin ada seorang karyawan memasuki 2 Laboratorium pada saat yang sama. Hal ini juga menjaga terhadap kemungkinan penyalahgunaan kunci password karyawan. Kemudian pula pengontrolan akan bisa jarak jauh (beberapa kilometer) untuk mengantisipasi jauhnya letak tiap-tiap pintu dan pusat pengontrolnya.

Dengan pembuatan Interface ini diharapkan dapat meningkatkan efisiensi perusahaan dalam mengontrol ketepatan dan keamanan ruangan-ruangan.

4. Penelaahan Study :

- Pemahaman Frequensi Shift Keying
- Pemahaman Teknik Interfacing pada PC
- Pemahaman pembuatan program dengan bahasa Assembly dan C
- Pemahaman cara penyimpanan data

5. Tujuan :

Membuat pengontrol pintu otomatis dengan menggunakan teknik FSK, yang diatur dengan menggunakan software untuk mengontrol keluar masuk karyawan dalam sebuah perusahaan.

6. Relevansi :

Diharapkan alat ini dapat mengefisiensikan dan mempermudah perusahaan dalam pengontrolan keluar masuk karyawan dalam perusahaan tersebut.

7. Langkah-langkah : 1. Studi Literatur

2. Perencanaan sistem
3. Pembuatan Hardware.
4. Pembuatan Program
5. Pengujian unjuk kerja alat dan program
6. Penulisan naskah tugas akhir

8. Jadwal Kerja :

No.	KEGIATAN	BULAN KE					
		I	II	III	IV	V	VI
1	Studi literatur						
2	Perencanaan sistem						
3.	Pembuatan Hardware						
4	Pembuatan Program						
5.	Pengujian unjuk kerja alat dan program						
6	Penulisan naskah tugas akhir						

RIWAYAT HIDUP PENULIS

Penulis dilahirkan pada tanggal 28 Mei 1969 di Balige, Sumatra Utara, yang merupakan tra keempat dari pasangan R. Bambang Suminang dan Etty Sutjiati, dan penulis diberi nama : URLAMBANG ATMOJO.

wayat Hidup Penulis :

1. SD Maria Fatima II Kaliwates, Jember, dijalani selama tahun 1975 - 1981
2. SMP Negeri I Surabaya, Genteng Kotamadya Surabaya, dijalani selama 1981 - 1984
3. SMA Negeri II Surabaya, Genteng Kotamadya Surabaya, dijalani selama 1984 -1987
4. DIII Politeknik Elektronika Program Studi Elektronika Industri, Universitas Brawijaya, dijalani selama 1987 -1990
5. Teknik Elektro ITS Surabaya, Bidang Studi Komputer dan Digital, dijalani selama 1994 - sekarang.

Sepenggal Kehidupan Penulis :

Selama menjadi mahasisiwa di kampus ITS, penulis melakukan berbagai macam kegiatan akademis. Salah satu kegiatan tersebut adalah menjadi asisten bagi mata kuliah bidang studi komputer dan digital pada Laboratorium Komputer dan Digital, B201. Penulis memulai kegiatan ini di mulai pada tahun 1990 dan masih berlanjut sampai sekarang saat penulis merancang dan menyelesaikan Tugas Akhir ini.